

SimFQT

0.1.3

Generated by Doxygen 1.8.1.2

Wed Aug 29 2012 11:42:35

Contents

1	SimFQT Documentation	1
1.1	Getting Started	2
1.2	SimFQT at SourceForge	2
1.3	SimFQT Development	2
1.4	External Libraries	2
1.5	Support SimFQT	3
1.6	About SimFQT	3
2	People	3
2.1	Project Admins (and Developers)	3
2.2	Retired Developers	3
2.3	Contributors	3
2.4	Distribution Maintainers	3
3	Coding Rules	3
3.1	Default Naming Rules for Variables	4
3.2	Default Naming Rules for Functions	4
3.3	Default Naming Rules for Classes and Structures	4
3.4	Default Naming Rules for Files	4
3.5	Default Functionality of Classes	4
4	Copyright and License	4
4.1	GNU LESSER GENERAL PUBLIC LICENSE	4
4.1.1	Version 2.1, February 1999	4
4.2	Preamble	5
4.3	TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION	6
4.3.1	NO WARRANTY	10
4.3.2	END OF TERMS AND CONDITIONS	10
4.4	How to Apply These Terms to Your New Programs	10
5	Documentation Rules	11
5.1	General Rules	11
5.2	File Header	11
5.3	Grouping Various Parts	12
6	Main features	12
6.1	Fare calculation	12
6.2	Fare rule engine	12
6.3	Fare retrieval	12
6.4	Other features	13

7	Make a Difference	13
8	Make a new release	13
8.1	Introduction	13
8.2	Initialisation	13
8.3	Release branch maintenance	14
8.4	Commit and publish the release branch	14
8.5	Create distribution packages	14
8.6	Upload the HTML documentation to SourceForge	14
8.7	Generate the RPM packages	15
8.8	Update distributed change log	15
8.9	Create the binary package, including the documentation	15
8.10	Upload the files to SourceForge	15
8.11	Make a new post	16
8.12	Send an email on the announcement mailing-list	16
9	Installation	16
9.1	Table of Contents	16
9.2	Fedora/RedHat Linux distributions	16
9.3	SimFQT Requirements	16
9.4	Basic Installation	17
9.5	Compilers and Options	18
9.6	Compiling For Multiple Architectures	18
9.7	Installation Names	18
9.8	Optional Features	19
9.9	Particular systems	20
9.10	Specifying the System Type	20
9.11	Sharing Defaults	21
9.12	Defining Variables	21
9.13	'cmake' Invocation	21
10	Linking with SimFQT	24
10.1	Table of Contents	24
10.2	Introduction	25
10.3	Dependencies	25
10.3.1	StdAir	25
10.4	Using the pkg-config command	25
10.5	Using the simfqt-config script	25
10.6	M4 macro for the GNU Autotools	25
10.7	Using SimFQT with dynamic linking	26

11 Test Rules	26
11.1 The Test File	26
11.2 The Reference File	26
11.3 Testing SimFQT Library	26
12 Users Guide	26
12.1 Table of Contents	27
12.2 Introduction	27
12.3 Get Started	27
12.3.1 Get the SimFQT library	27
12.3.2 Build the SimFQT project	27
12.3.3 Run the Tests	28
12.3.4 Install the SimFQT Project (Binaries, Documentation)	28
12.4 Input file of SimFQT Project	28
12.5 The fare quoting BOM Tree	29
12.5.1 Build of the fare quoting BOM tree	29
12.5.2 Display of the fare quoting BOM tree	30
12.5.3 Structure of the fare quoting BOM tree	30
12.6 The fare quoting procedure	31
12.6.1 Instantiate the default booking request	31
12.6.2 Instantiate the default travel solution list	31
12.6.3 Fare Quoting a list of travel solution	31
12.7 Error Messages	31
12.7.1 Fare input file not found	32
12.7.2 Fare input file can not be parsed	32
12.7.3 Error Messages for missing fare rules	32
13 Supported Systems	33
13.1 Table of Contents	33
13.2 Introduction	34
13.3 SimFQT 3.10.x	34
13.3.1 Linux Systems	34
13.3.2 Windows Systems	38
13.3.3 Unix Systems	40
14 SimFQT Supported Systems (Previous Releases)	40
14.1 SimFQT 3.9.1	40
14.2 SimFQT 3.9.0	40
14.3 SimFQT 3.8.1	40
15 Tutorials	41

15.1 Table of Contents	41
15.2 Preparing the SimFQT Project for Development	41
15.3 Your first fareQuote	41
15.3.1 Summary of the different steps	41
15.3.2 Result of the Batch Program	41
15.4 Fare quoting with an input file	42
15.4.1 How to build a fare input file?	42
15.4.2 Building the BOM tree with an input file	44
15.4.3 Result of the Batch Program	44
16 Command-Line Test to Demonstrate How To Test the SimFQT Project	44
17 Namespace Index	47
17.1 Namespace List	48
18 Class Index	48
18.1 Class Hierarchy	48
19 Class Index	50
19.1 Class List	50
20 File Index	52
20.1 File List	52
21 Namespace Documentation	52
21.1 SIMFQT Namespace Reference	53
21.1.1 Typedef Documentation	53
21.1.2 Variable Documentation	54
21.2 SIMFQT::FareParserHelper Namespace Reference	54
21.2.1 Variable Documentation	54
21.3 stdair Namespace Reference	55
21.3.1 Detailed Description	56
22 Class Documentation	56
22.1 SIMFQT::AirlineNotFoundException Class Reference	56
22.1.1 Detailed Description	56
22.1.2 Constructor & Destructor Documentation	56
22.2 SIMFQT::AirportPairNotFoundException Class Reference	56
22.2.1 Detailed Description	57
22.2.2 Constructor & Destructor Documentation	57
22.3 CmdAbstract Class Reference	57
22.4 SIMFQT::FareParserHelper::doEndFare Struct Reference	57
22.4.1 Detailed Description	58

22.4.2	Constructor & Destructor Documentation	58
22.4.3	Member Function Documentation	58
22.4.4	Member Data Documentation	58
22.5	FacServiceAbstract Class Reference	59
22.6	SIMFQT::FacSimfqtServiceContext Class Reference	59
22.6.1	Detailed Description	59
22.6.2	Constructor & Destructor Documentation	60
22.6.3	Member Function Documentation	60
22.7	SIMFQT::FareFileParsingFailedException Class Reference	60
22.7.1	Detailed Description	61
22.7.2	Constructor & Destructor Documentation	61
22.8	SIMFQT::FareFilePath Class Reference	61
22.8.1	Detailed Description	61
22.8.2	Constructor & Destructor Documentation	61
22.9	SIMFQT::FareInputFileNotFoundException Class Reference	62
22.9.1	Detailed Description	62
22.9.2	Constructor & Destructor Documentation	62
22.10	SIMFQT::FareParser Class Reference	62
22.10.1	Detailed Description	62
22.10.2	Member Function Documentation	63
22.11	SIMFQT::FareQuoter Class Reference	63
22.11.1	Detailed Description	63
22.11.2	Friends And Related Function Documentation	63
22.12	SIMFQT::FareRuleFileParser Class Reference	63
22.12.1	Detailed Description	64
22.12.2	Constructor & Destructor Documentation	64
22.12.3	Member Function Documentation	64
22.13	SIMFQT::FareRuleGenerator Class Reference	64
22.13.1	Detailed Description	65
22.13.2	Friends And Related Function Documentation	65
22.14	SIMFQT::FareParserHelper::FareRuleParser< Iterator > Struct Template Reference	65
22.14.1	Detailed Description	67
22.14.2	Constructor & Destructor Documentation	67
22.14.3	Member Data Documentation	68
22.15	SIMFQT::FareRuleStruct Struct Reference	71
22.15.1	Detailed Description	72
22.15.2	Constructor & Destructor Documentation	72
22.15.3	Member Function Documentation	72
22.15.4	Member Data Documentation	78
22.16	SIMFQT::FeaturesNotFoundedException Class Reference	78

22.16.1 Detailed Description	79
22.16.2 Constructor & Destructor Documentation	79
22.17FileNotFoundException Class Reference	79
22.18SIMFQT::FlightDateNotFoundException Class Reference	79
22.18.1 Detailed Description	80
22.18.2 Constructor & Destructor Documentation	80
22.19SIMFQT::FlightTimeNotFoundException Class Reference	80
22.19.1 Detailed Description	80
22.19.2 Constructor & Destructor Documentation	80
22.20grammar Class Reference	80
22.21InputFilePath Class Reference	81
22.22ObjectNotFoundException Class Reference	81
22.23SIMFQT::FareParserHelper::ParserSemanticAction Struct Reference	81
22.23.1 Detailed Description	82
22.23.2 Constructor & Destructor Documentation	82
22.23.3 Member Data Documentation	82
22.24ParsingFileFailedException Class Reference	83
22.25SIMFQT::PosOrChannelNotFoundException Class Reference	83
22.25.1 Detailed Description	84
22.25.2 Constructor & Destructor Documentation	84
22.26SIMFQT::QuotingException Class Reference	84
22.26.1 Detailed Description	84
22.27RootException Class Reference	84
22.28ServiceAbstract Class Reference	85
22.29SIMFQT::SIMFQT_Service Class Reference	85
22.29.1 Detailed Description	85
22.29.2 Constructor & Destructor Documentation	85
22.29.3 Member Function Documentation	86
22.30SIMFQT::SIMFQT_ServiceContext Class Reference	89
22.30.1 Detailed Description	89
22.30.2 Friends And Related Function Documentation	89
22.31SIMFQT::FareParserHelper::storeAdvancePurchase Struct Reference	90
22.31.1 Detailed Description	90
22.31.2 Constructor & Destructor Documentation	90
22.31.3 Member Function Documentation	90
22.31.4 Member Data Documentation	91
22.32SIMFQT::FareParserHelper::storeAirlineCode Struct Reference	91
22.32.1 Detailed Description	91
22.32.2 Constructor & Destructor Documentation	91
22.32.3 Member Function Documentation	92

22.32.4 Member Data Documentation	92
22.33SIMFQT::FareParserHelper::storeCabinCode Struct Reference	92
22.33.1 Detailed Description	93
22.33.2 Constructor & Destructor Documentation	93
22.33.3 Member Function Documentation	93
22.33.4 Member Data Documentation	93
22.34SIMFQT::FareParserHelper::storeChangeFees Struct Reference	93
22.34.1 Detailed Description	94
22.34.2 Constructor & Destructor Documentation	94
22.34.3 Member Function Documentation	94
22.34.4 Member Data Documentation	94
22.35SIMFQT::FareParserHelper::storeChannel Struct Reference	95
22.35.1 Detailed Description	95
22.35.2 Constructor & Destructor Documentation	95
22.35.3 Member Function Documentation	95
22.35.4 Member Data Documentation	96
22.36SIMFQT::FareParserHelper::storeClass Struct Reference	96
22.36.1 Detailed Description	96
22.36.2 Constructor & Destructor Documentation	96
22.36.3 Member Function Documentation	97
22.36.4 Member Data Documentation	97
22.37SIMFQT::FareParserHelper::storeDateRangeEnd Struct Reference	97
22.37.1 Detailed Description	98
22.37.2 Constructor & Destructor Documentation	98
22.37.3 Member Function Documentation	98
22.37.4 Member Data Documentation	98
22.38SIMFQT::FareParserHelper::storeDateRangeStart Struct Reference	98
22.38.1 Detailed Description	99
22.38.2 Constructor & Destructor Documentation	99
22.38.3 Member Function Documentation	99
22.38.4 Member Data Documentation	99
22.39SIMFQT::FareParserHelper::storeDestination Struct Reference	100
22.39.1 Detailed Description	100
22.39.2 Constructor & Destructor Documentation	100
22.39.3 Member Function Documentation	100
22.39.4 Member Data Documentation	101
22.40SIMFQT::FareParserHelper::storeEndRangeTime Struct Reference	101
22.40.1 Detailed Description	101
22.40.2 Constructor & Destructor Documentation	102
22.40.3 Member Function Documentation	102

22.40.4 Member Data Documentation	102
22.41 SIMFQT::FareParserHelper::storeFare Struct Reference	102
22.41.1 Detailed Description	103
22.41.2 Constructor & Destructor Documentation	103
22.41.3 Member Function Documentation	103
22.41.4 Member Data Documentation	103
22.42 SIMFQT::FareParserHelper::storeFareId Struct Reference	104
22.42.1 Detailed Description	104
22.42.2 Constructor & Destructor Documentation	104
22.42.3 Member Function Documentation	104
22.42.4 Member Data Documentation	104
22.43 SIMFQT::FareParserHelper::storeMinimumStay Struct Reference	105
22.43.1 Detailed Description	105
22.43.2 Constructor & Destructor Documentation	105
22.43.3 Member Function Documentation	105
22.43.4 Member Data Documentation	106
22.44 SIMFQT::FareParserHelper::storeNonRefundable Struct Reference	106
22.44.1 Detailed Description	107
22.44.2 Constructor & Destructor Documentation	107
22.44.3 Member Function Documentation	107
22.44.4 Member Data Documentation	107
22.45 SIMFQT::FareParserHelper::storeOrigin Struct Reference	107
22.45.1 Detailed Description	108
22.45.2 Constructor & Destructor Documentation	108
22.45.3 Member Function Documentation	108
22.45.4 Member Data Documentation	108
22.46 SIMFQT::FareParserHelper::storePOS Struct Reference	109
22.46.1 Detailed Description	109
22.46.2 Constructor & Destructor Documentation	109
22.46.3 Member Function Documentation	109
22.46.4 Member Data Documentation	110
22.47 SIMFQT::FareParserHelper::storeSaturdayStay Struct Reference	110
22.47.1 Detailed Description	110
22.47.2 Constructor & Destructor Documentation	110
22.47.3 Member Function Documentation	111
22.47.4 Member Data Documentation	111
22.48 SIMFQT::FareParserHelper::storeStartRangeTime Struct Reference	111
22.48.1 Detailed Description	112
22.48.2 Constructor & Destructor Documentation	112
22.48.3 Member Function Documentation	112

22.48.4 Member Data Documentation	112
22.49SIMFQT::FareParserHelper::storeTripType Struct Reference	112
22.49.1 Detailed Description	113
22.49.2 Constructor & Destructor Documentation	113
22.49.3 Member Function Documentation	113
22.49.4 Member Data Documentation	113
22.50StructAbstract Class Reference	114
23 File Documentation	114
23.1 doc/local/authors.doc File Reference	114
23.2 doc/local/codingrules.doc File Reference	114
23.3 doc/local/copyright.doc File Reference	114
23.4 doc/local/documentation.doc File Reference	114
23.5 doc/local/features.doc File Reference	114
23.6 doc/local/help_wanted.doc File Reference	114
23.7 doc/local/howto_release.doc File Reference	114
23.8 doc/local/index.doc File Reference	114
23.9 doc/local/installation.doc File Reference	114
23.10doc/local/linking.doc File Reference	114
23.11doc/local/test.doc File Reference	114
23.12doc/local/users_guide.doc File Reference	114
23.13doc/local/verification.doc File Reference	114
23.14doc/tutorial/tutorial.doc File Reference	115
23.15simfqt/basic/BasConst.cpp File Reference	115
23.16BasConst.cpp	115
23.17simfqt/basic/BasConst_General.hpp File Reference	115
23.18BasConst_General.hpp	115
23.19simfqt/basic/BasConst_SIMFQT_Service.hpp File Reference	115
23.20BasConst_SIMFQT_Service.hpp	116
23.21simfqt/batches/simfqt_parseFareRules.cpp File Reference	116
23.21.1 Typedef Documentation	117
23.21.2 Function Documentation	117
23.21.3 Variable Documentation	117
23.22simfqt_parseFareRules.cpp	118
23.23simfqt/bom/FareRuleStruct.cpp File Reference	120
23.24FareRuleStruct.cpp	121
23.25simfqt/bom/FareRuleStruct.hpp File Reference	122
23.26FareRuleStruct.hpp	122
23.27simfqt/command/FareParser.cpp File Reference	126
23.28FareParser.cpp	126

23.29simfqt/command/FareParser.hpp File Reference	127
23.30FareParser.hpp	127
23.31simfqt/command/FareParserHelper.cpp File Reference	127
23.32FareParserHelper.cpp	128
23.33simfqt/command/FareParserHelper.hpp File Reference	137
23.34FareParserHelper.hpp	138
23.35simfqt/command/FareQuoter.cpp File Reference	140
23.36FareQuoter.cpp	141
23.37simfqt/command/FareQuoter.hpp File Reference	149
23.38FareQuoter.hpp	149
23.39simfqt/command/FareRuleGenerator.cpp File Reference	150
23.40FareRuleGenerator.cpp	151
23.41simfqt/command/FareRuleGenerator.hpp File Reference	154
23.42FareRuleGenerator.hpp	154
23.43simfqt/config/simfqt-paths.hpp File Reference	155
23.43.1 Macro Definition Documentation	155
23.44simfqt-paths.hpp	157
23.45simfqt/factory/FacSimfqtServiceContext.cpp File Reference	157
23.46FacSimfqtServiceContext.cpp	157
23.47simfqt/factory/FacSimfqtServiceContext.hpp File Reference	158
23.48FacSimfqtServiceContext.hpp	158
23.49simfqt/service/SIMFQT_Service.cpp File Reference	159
23.50SIMFQT_Service.cpp	159
23.51simfqt/service/SIMFQT_ServiceContext.cpp File Reference	164
23.52SIMFQT_ServiceContext.cpp	164
23.53simfqt/service/SIMFQT_ServiceContext.hpp File Reference	165
23.54SIMFQT_ServiceContext.hpp	165
23.55simfqt/SIMFQT_Service.hpp File Reference	166
23.56SIMFQT_Service.hpp	167
23.57simfqt/SIMFQT_Types.hpp File Reference	168
23.58SIMFQT_Types.hpp	169
23.59simfqt/ui/cmdline/simfqt.cpp File Reference	170
23.60simfqt.cpp	170
23.61test/simfqt/FQTestSuite.cpp File Reference	184
23.62FQTestSuite.cpp	184

1 SimFQT Documentation

1.1 Getting Started

- [Main features](#)
- [Installation](#)
- [Linking with SimFQT](#)
- [Users Guide](#)
- [Tutorials](#)
- [Copyright and License](#)
- [Make a Difference](#)
- [Make a new release](#)
- [People](#)

1.2 SimFQT at SourceForge

- [Project page](#)
- [Download SimFQT](#)
- [Open a ticket for a bug or feature](#)
- [Mailing lists](#)
- [Forums](#)
 - [Discuss about Development issues](#)
 - [Ask for Help](#)
 - [Discuss SimFQT](#)

1.3 SimFQT Development

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

1.4 External Libraries

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

1.5 Support SimFQT

1.6 About SimFQT

SimFQT is a C++ project of airline pricing classes and functions, mainly targeting simulation purposes. [N](#)

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

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

SimFQT 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 SimFQT 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 SimFQT.

2 People

2.1 Project Admins (and Developers)

- Gabrielle Sabatier gsabatier@users.sourceforge.net ([N](#))
- Denis Arnaud denis_arnaud@users.sourceforge.net ([N](#))
- Anh Quan Nguyen quannaus@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

Copyright (C) 1991, 1999 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

4.2 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages—typically libraries—of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

4.3 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

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

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has


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

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

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

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

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

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

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

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

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

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

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

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

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

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

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

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

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
- b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
- c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
- d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
- e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

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

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

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

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
- b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

1. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and

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

1. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.
1. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.
1. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

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

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

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

1. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
1. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

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

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

4.3.1 NO WARRANTY

1. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
1. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

4.3.2 END OF TERMS AND CONDITIONS

4.4 How to Apply These Terms to Your New Programs

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>
```

```
This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.
```

```
This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.
```

```
You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
library 'Frob' (a library for tweaking knobs) written by James Random Hacker.
```

```
<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice
```

That's all there is to it!

Source

5 Documentation Rules

5.1 General Rules

All classes in SimFQT 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 SimFQT 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.
*
* -----
*
* SimFQT - C++ Standard Airline IT Object Library
*
* Copyright (C) 2009-2010 (\see authors file for a list of contributors)
*
* \see copyright file for license information
*
* -----
*/

```

5.3 Grouping Various Parts

All functions must be added to a Doxygen group in order to appear in the documentation. The following code example defines the group 'my_group':

```

/*!
* \defgroup my_group Brief description of the group here
*
* Detailed description of the group here
*/

```

The following example shows how to document the function `myFunction` and how to add it to the group `my_group`:

```

/*!
* \brief Brief description of myFunction here
* \ingroup my_group
*
* Detailed description of myFunction here
*
* \param[in] param1 Description of \a param1 here
* \param[in] param2 Description of \a param2 here
* \return Description of the return value here
*/
TYPE3 myFunction(TYPE1 param1, TYPE2 &param2);

```

6 Main features

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

6.1 Fare calculation

- Calculation of fare from statistics on tickets/coupons

6.2 Fare rule engine

- Fare rules: storage, engine, management

6.3 Fare retrieval

- Retrieval of fares 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 SimFQT can do for you. Ask what you can do for SimFQT.

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

- Start using SimFQT
- Tell your friends about SimFQT 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 SimFQT discussion forums on SourceForge. If you know the answer to a question, help others to overcome their SimFQT problems.
- Help us to improve our algorithms. If you know of a better way (e.g., that is faster or requires less memory) to implement some of our algorithms, then let us know.
- Help to port SimFQT to new platforms. If you manage to compile SimFQT on a new platform, then tell how you did it.
- Send us your code. If you have a good SimFQT compatible code, which you can release under the LGPL, and you think it should be included in SimFQT, then send it to the community.
- Become an SimFQT developer. Send us an e-mail and tell what you can do for SimFQT.

8 Make a new release

8.1 Introduction

This document describes briefly the recommended procedure of releasing a new version of SimFQT 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://simfqt.git.sourceforge.net/gitroot/simfqt/simfqt simfqtgit
cd simfqtgit
git checkout trunk
```

8.3 Release branch maintenance

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

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

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

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

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

```
vi NEWS
vi ChangeLog
vi simfqt.spec
```

8.4 Commit and publish the release branch

Commit the new release:

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

8.5 Create distribution packages

Create the distribution packages using the following command:

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

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

8.6 Upload the HTML documentation to SourceForge

In order to update the Web site files, either:

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


```
cd ~/dev/sim/simfqtgit/build
git checkout releases
rsync -aiv ${INSTALL_BASEDIR}/simfqt-0.5.0/share/doc/simfqt-0.5.0/html/ \
  your_sf_user,simfqt@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](#).

8.7 Generate the RPM packages

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

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

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

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

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 [SimFQT'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:

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

The output binary package will be named, for instance, `simfqt-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 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](#).

8.12 Send an email on the announcement mailing-list

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

9 Installation

9.1 Table of Contents

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

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 simfqt-devel simfqt-doc
```

RPM packages can also be available on the [SourceForge download site](#).

9.3 SimFQT Requirements

SimFQT 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 SimFQT.

9.4 Basic Installation

Briefly, the shell commands `./cmake .. && make install` should configure, build, and install this package. The following more-detailed instructions are generic; see the `'README'` file for instructions specific to this package. Some packages provide this `'INSTALL'` file but do not implement all of the features documented below. The lack of an optional feature in a given package is not necessarily a bug. More recommendations for GNU packages can be found in the info page corresponding to "Makefile Conventions: (standards)Makefile Conventions".

The `'cmake'` shell script attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create a `'Makefile'` in each directory of the package. It may also create one or more `'.h'` files containing system-dependent definitions. Finally, it creates a `'CMakeCache.txt'` cache file that you can refer to in the future to recreate the current configuration, and a file `'CMakeFiles'` containing compiler output (useful mainly for debugging `'cmake'`).

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

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

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

See Also

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

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

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

The 'cmake' script produces an output like this:

```
-- Requires Git without specifying any version
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/simfqt-0.5.0 \
-DWITH_STDAIR_PREFIX=/home/user/dev/deliveries/stdair-stable \
-DLIB_SUFFIX=64 -DCMAKE_BUILD_TYPE:String=Debug -DINSTALL_DOC:BOOL=ON ..
-- Current Git revision name: 0e31d63879056d26f01eb09757d232d247c42164 trunk
-- Requires Boost-1.41
-- Found Boost version: 1.44.0
-- Requires Readline without specifying any version
-- Found Readline version: 6.1
-- Requires MySQL without specifying any version
-- Using mysql-config: /usr/bin/mysql_config
-- Found MySQL version: 5.1.56
-- Requires SOCI-3.0
-- Using soci-config: /usr/bin/soci-config
-- SOCI headers are buried
-- Found SOCI with MySQL back-end support version: 3.0.0
-- Requires StdAir-0.35
-- Found StdAir version: 99.99.99
-- Requires Doxygen without specifying any version
-- Found Doxygen version: 1.7.4
-- Had to set the linker language for 'simfqtlib' to CXX
-- Test 'FQTTestSuite' to be built with 'FQTTestSuite.cpp'
--
-- =====
-- -----
-- ---      Project Information      ---
-- -----
-- PROJECT_NAME ..... : simfqt
-- PACKAGE_PRETTY_NAME ..... : SimFQT
-- PACKAGE ..... : simfqt
-- PACKAGE_NAME ..... : SIMFQT
-- PACKAGE_BRIEF ..... : C++ Simulated Fare Quote System Library
-- PACKAGE_VERSION ..... : 99.99.99
-- GENERIC_LIB_VERSION ..... : 99.99.99
-- GENERIC_LIB_SOVERSION ..... : 99.99
--
-- -----
-- ---      Build Configuration      ---
-- -----
-- Modules to build ..... : simfqt
-- Libraries to build/install ..... : simfqtlib
-- Binaries to build/install ..... : simfqt;fareQuote
-- Modules to test ..... : simfqt
-- Binaries to test ..... : FQTTestSuitetst
--
-- * Module ..... : simfqt
--   + Layers to build ..... : .;basic;bom;factory;command;service
--   + Dependencies on other layers :
--   + Libraries to build/install . : simfqtlib
--   + Executables to build/install : simfqt;fareQuote
--   + Tests to perform ..... : FQTTestSuitetst
--
-- BUILD_SHARED_LIBS ..... : ON
-- CMAKE_BUILD_TYPE ..... : Debug
-- * CMAKE_C_FLAGS ..... :
-- * CMAKE_CXX_FLAGS ..... : -Wall -Werror
-- * BUILD_FLAGS ..... :
-- * COMPILE_FLAGS ..... :
-- CMAKE_MODULE_PATH ..... : /home/localoriuser/dev/sim/simfqt/simfqtgit/config/
```



```

-- CMAKE_INSTALL_PREFIX ..... : /home/localoriuser/dev/deliveries/simfqt-99.99.99
--
-- * Doxygen:
--   - DOXYGEN_VERSION ..... : 1.7.4
--   - DOXYGEN_EXECUTABLE ..... : /usr/bin/doxygen
--   - DOXYGEN_DOT_EXECUTABLE ..... : DOXYGEN_DOT_EXECUTABLE-NOTFOUND
--   - DOXYGEN_DOT_PATH ..... :
--
-- -----
-- --- Installation Configuration ---
-- -----
-- INSTALL_LIB_DIR ..... : /home/localoriuser/dev/deliveries/simfqt-99.99.99/lib
-- INSTALL_BIN_DIR ..... : /home/localoriuser/dev/deliveries/simfqt-99.99.99/bin
-- INSTALL_INCLUDE_DIR ..... : /home/localoriuser/dev/deliveries/simfqt-99.99.99/include
-- INSTALL_DATA_DIR ..... : /home/localoriuser/dev/deliveries/simfqt-99.99.99/share
-- INSTALL_SAMPLE_DIR ..... : /home/localoriuser/dev/deliveries/simfqt-99.99.99/share/simfqt/samples
-- INSTALL_DOC ..... : ON
--
-- -----
-- --- Packaging Configuration ---
-- -----
-- CPACK_PACKAGE_CONTACT ..... : Denis Arnaud <denis_arnaud - at - users dot sourceforge dot net>
-- CPACK_PACKAGE_VENDOR ..... : Denis Arnaud
-- CPACK_PACKAGE_VERSION ..... : 99.99.99
-- CPACK_PACKAGE_DESCRIPTION_FILE . : /home/localoriuser/dev/sim/simfqt/simfqtgit/README
-- CPACK_RESOURCE_FILE_LICENSE .... : /home/localoriuser/dev/sim/simfqt/simfqtgit/COPYING
-- CPACK_GENERATOR ..... : TBZ2
-- CPACK_DEBIAN_PACKAGE_DEPENDS ... :
-- CPACK_SOURCE_GENERATOR ..... : TBZ2;TGZ
-- CPACK_SOURCE_PACKAGE_FILE_NAME . : simfqt-99.99.99
--
-- -----
-- --- External libraries ---
-- -----
--
-- * Boost:
--   - Boost_VERSION ..... : 104400
--   - Boost_LIB_VERSION ..... : 1_44
--   - Boost_HUMAN_VERSION ..... : 1.44.0
--   - Boost_INCLUDE_DIRS ..... : /usr/include
--   - Boost required components .. : program_options;date_time;iostreams;serialization;filesystem;unit_test_f
--   - Boost required libraries ... : optimized;/usr/lib/libboost_iostreams-mt.so;debug;/usr/lib/libboost_iost
--
-- * Readline:
--   - READLINE_VERSION ..... : 6.1
--   - READLINE_INCLUDE_DIR ..... : /usr/include
--   - READLINE_LIBRARY ..... : /usr/lib/libreadline.so
--
-- * MySQL:
--   - MYSQL_VERSION ..... : 5.1.56
--   - MYSQL_INCLUDE_DIR ..... : /usr/include/mysql
--   - MYSQL_LIBRARIES ..... : /usr/lib/mysql/libmysqlclient_r.so
--
-- * SOCI:
--   - SOCI_VERSION ..... : 3.0.0
--   - SOCI_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_MYSQL_INCLUDE_DIR ..... : /usr/include/soci
--   - SOCI_LIBRARIES ..... : /usr/lib/libsoci_core.so
--   - SOCI_MYSQL_LIBRARIES ..... : /usr/lib/libsoci_mysql.so
--
-- * StdAir:
--   - STDAIR_VERSION ..... : 99.99.99
--   - STDAIR_BINARY_DIRS ..... : /home/localoriuser/dev/deliveries/stdair-0.3.0/bin
--   - STDAIR_EXECUTABLES ..... : stdair
--   - STDAIR_LIBRARY_DIRS ..... : /home/localoriuser/dev/deliveries/stdair-0.3.0/lib
--   - STDAIR_LIBRARIES ..... : stdairlib;stdairuiclib
--   - STDAIR_INCLUDE_DIRS ..... : /home/localoriuser/dev/deliveries/stdair-0.3.0/include
--   - STDAIR_SAMPLE_DIR ..... : /home/localoriuser/dev/deliveries/stdair-0.3.0/share/stdair/samples
--
-- Change a value with: cmake -D<Variable>=<Value>
-- =====
--
-- Configuring done

```

```
-- Generating done
-- Build files have been written to: /home/localoriuser/dev/sim/simfqt/simfqtgit/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_simfqt
[ 90%] Built target simfqtlib
[100%] Built target FQTTestSuitetst
Test project /home/localoriuser/dev/sim/simfqt/simfqtgit/build/test/simfqt
  Start 1: FQTTestSuitetst
1/1 Test #1: FQTTestSuitetst ..... Passed    0.43 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) = 0.47 sec
[100%] Built target check_simfqtst
[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/simfqtgit
rm -rf build && mkdir build
cd build
```

to remove everything.

10 Linking with SimFQT

10.1 Table of Contents

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

10.2 Introduction

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

10.3 Dependencies

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

10.3.1 StdAir

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

10.4 Using the pkg-config command

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

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

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

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

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

10.5 Using the simfqt-config script

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

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

```
simfqt-config --help
```

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

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

10.6 M4 macro for the GNU Autotools

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

- `'SIMFQT_VERSION'` (e.g., defined to 0.2.0)
- `'SIMFQT_CFLAGS'` (e.g., defined to `'-I${prefix}/include'`)
- `'SIMFQT_LIBS'` (e.g., defined to `'-L${prefix}/lib -lsimfqt'`)

10.7 Using SimFQT with dynamic linking

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

11 Test Rules

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

11.1 The Test File

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

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

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

11.2 The Reference File

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

11.3 Testing SimFQT Library

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

```
% make check
```

after successful compilation of the SimFQT library.

12 Users Guide

12.1 Table of Contents

- [Introduction](#)
- [Get Started](#)
 - [Get the SimFQT library](#)
 - [Build the SimFQT project](#)
 - [Run the Tests](#)
 - [Install the SimFQT Project \(Binaries, Documentation\)](#)
- [Input file of SimFQT Project](#)
- [The fare quoting BOM Tree](#)
 - [Build of the fare quoting BOM tree](#)
 - [Display of the fare quoting BOM tree](#)
 - [Structure of the fare quoting BOM tree](#)
- [The fare quoting procedure](#)
 - [Instantiate the default booking request](#)
 - [Instantiate the default travel solution list](#)
 - [Fare Quoting a list of travel solution](#)
- [Error Messages](#)
 - [Fare input file not found](#)
 - [Fare input file can not be parsed](#)
 - [Error Messages for missing fare rules](#)

12.2 Introduction

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

12.3 Get Started

12.3.1 Get the SimFQT library

Clone locally the full [Git project](#):

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

12.3.2 Build the SimFQT project

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

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

12.3.3 Run the Tests

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

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

As a result, you should obtain a similar report:

```
[ 0%] Built target hdr_cfg_simfqt
[ 90%] Built target simfqtlib
[100%] Built target FQTTestSuitetst
Test project /home/localoriuser/dev/sim/simfqt/simfqtgit/build/test/simfqt
Start 1: FQTTestSuitetst
1/1 Test #1: FQTTestSuitetst ..... Passed    0.15 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) = 0.16 sec
[100%] Built target check_simfqtst
[100%] Built target check
```

12.3.4 Install the SimFQT Project (Binaries, Documentation)

After the step [Build the SimFQT project](#), to install the library and its header files, type:

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

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

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

To generate the SimFQT project documentation, the commands are:

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

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

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

12.4 Input file of SimFQT Project

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

```
// Fares: fare ID; OriginCity; DestinationCity; TripType; DateRangeStart;
DateRangeEnd; DepartureTimeRangeStart; DepartureTimeRangeEnd; POS; CabinCode;
Channel; AdvancePurchase; SaturdayNight; ChangeFees; NonRefundable; MinimumStay; Price;
nb Segments
// Segment: AirlineCode; Class;
1; SIN; BKK; OW; 2010-01-15; 2010-12-31; 00:00; 23:59; SIN; Y; IN; 7; T; T; T;
3; 150.0; SQ; Y;
2; SIN; BKK; OW; 2010-01-15; 2010-12-31; 00:00; 23:59; BKK; Y; IN; 7; T; T; T;
3; 150.0; SQ; Y;
3; SIN; HND; OW; 2010-01-15; 2010-12-31; 00:00; 23:59; SIN; Y; IN; 7; T; T; T;
3; 150.0; SQ; Y;
4; SIN; HND; OW; 2010-01-15; 2010-12-31; 00:00; 23:59; HND; Y; IN; 7; T; T; T;
```

```

3; 150.0; SQ; Y;
5; SIN; HND; OW; 2010-01-15; 2010-12-31; 00:00; 23:59; ROW; Y; IN; 7; T; T; T;
3; 150.0; SQ; Y;
6; SIN; BKK; OW; 2010-01-15; 2010-12-31; 00:00; 23:59; SIN; Y; IF; 7; T; T; T;
3; 150.0; SQ; Y;
7; SIN; BKK; OW; 2010-01-15; 2010-12-31; 00:00; 23:59; BKK; Y; IF; 7; T; T; T;
3; 150.0; SQ; Y;
8; SIN; HND; OW; 2010-01-15; 2010-12-31; 00:00; 23:59; SIN; Y; IF; 7; T; T; T;
3; 150.0; SQ; Y;
9; SIN; HND; OW; 2010-01-15; 2010-12-31; 00:00; 23:59; HND; Y; IF; 7; T; T; T;
3; 150.0; SQ; Y;
10; SIN; HND; OW; 2010-01-15; 2010-12-31; 00:00; 23:59; ROW; Y; IF; 7; T; T; T;
3; 150.0; SQ; Y;

```

Each line represents a fare rule (see [SIMFQT::FareRuleStruct](#)), i.e., each line tells us the price a customer will be asked according to a lot of criteria such as:

- the origin and destination of his travel (for instance from Singapour to Bangkok in the first fare rule).
- the type of his travel, i.e. one-way "OW" or round trip "RT".
- the date and time he is willing to travel (each fare rule has a date range and a time range of validity).
- the place where he is buying the ticket, i.e. the point of sale.
- his preferred cabin.
- the channel of the booking described by a two letters code: direct(D)/indirect(I) and online(N)/offline(F).
- the date when he wants to buy the ticket, i.e. the advanced purchase required in number of days.
- the saturday night stay option, i.e. is he staying a saturday night between his inbound trip and his outbound one? "T" stands for true and "F" stands for false.
- the change fees option, i.e. are there fees to change his ticket? "T" stands for true and "F" stands for false.
- the refundable criterion, i.e. is the ticket refundable? "T" stands for true and "F" stands for false.
- the number of days he is willing to stay at the destination location (each fare rule has a minimum stay requirement in number of days).

Some fare input examples (including the example above named fare01.csv) are given in the `stdair::samples` directory.

12.5 The fare quoting BOM Tree

The Fare Quoting Business Object Model (BOM) tree is a structure permitting to store all the [SIMFQT::FareRuleStruct](#) objects of the simulation. That is why, the BOM tree is built parsing the fare file containing all the fare rules (as described in the previous section [Input file of SimFQT Project](#)). For convenience and first use of SimFQT (the input fare file building can be long and heavy), SimFQT API enables to build a small default BOM tree.

12.5.1 Build of the fare quoting BOM tree

First, a BOM root object (i.e., a root for all the classes in the project) is instantiated by the `stdair::STDAIR_ServiceContext` context object, when the `stdair::STDAIR_Service` is itself instantiated, that is to say during the instantiation of the `simfqt::SIMFQT_Service` object. The corresponding type (class) `stdair::BomRoot` is defined in the `StdAir` library.

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

```
void buildSampleBom();
```

or can be constructed using the fare dump file described above thanks to the `simfqt::SIMFQT_Service::parseAndLoad (const stdair::Filename_T&) method:`

```
void parseAndLoad (const FareFilePath& iFareFilename);
```

12.5.2 Display of the fare quoting BOM tree

The fare quoting BOM tree can be displayed as done in the `batches::simfqt.cpp` program:

When the default bom tree is used (`-b` option of the main program `simfqt.cpp`), the fare quoting BOM tree display should look like:

```
=====
BomRoot:  -- ROOT --
=====
+++++
AirportPair: LHR, SYD
+++++
-----
DatePeriod: [2011-Jan-15/2011-Dec-30]
-----
*****
PosChannel: LHR,DN
*****
-----
TimePeriod: 00:00:00-23:00:00
-----
-----
Fare-Features: RT -- 0-1-1-1-0
-----
-----
AirlineClassList: BA Y
-----
```

Here the fare quoting BOM tree is just composed of one fare rule.

12.5.3 Structure of the fare quoting BOM tree

As one can guess looking at the BOM tree display above, the tree is constructed as follow:

- At the top of the tree, we find a `stdair::BomRoot` object (i.e., a root for all the classes in the project).
- Just under the root, at the first level, we find `stdair::AirportPair` objects (i.e., all the possible combinations of origin-destination). In the instance above, the only combination possible is from London to Sydney.
- At the next level, under a particular `stdair::AirportPair`, we find all the date periods of the fare rules applicable for this origin-destination.
- Then, under a particular `stdair::DatePeriod`, we find all the possible combinations of point-of-sale and channel applicable.
- Under a particular `stdair::PosChannel` object, we have the corresponding `stdair::TimePeriod` objects.
- At the next-to-last level, we have `stdair::FareFeatures` objects, that is to say the trip type, the advanced purchase and stay duration required, ...
- Finally we find the code of the airline publishing the current fare rule and the applicable class code.

12.6 The fare quoting procedure

The project SimFQT aims at fare quoting a list of **travel solutions** corresponding to a **booking request**. The fare quoter looks for all the fare rules matching a travel solution: when a fare rule matches, it creates a **fare option** object and adds this object to the current travel solution.

A few steps:

- [Instantiate the default booking request](#)
- [Instantiate the default travel solution list](#)
- [Fare Quoting a list of travel solution](#)

12.6.1 Instantiate the default booking request

A default booking request can be built using the `simfqt::SIMFQT_Service::buildBookingRequest` method:

```
stdair::BookingRequestStruct buildBookingRequest(const bool isForCRS =
    false);
```

12.6.2 Instantiate the default travel solution list

In the following sample, a list of travel solutions is given as input/output parameter of the `simfqt::SIMFQT_Service::buildSampleTravelSolutions` method:

```
void buildSampleTravelSolutions (stdair::TravelSolutionList_T&);
```

12.6.3 Fare Quoting a list of travel solution

Once a booking request, its corresponding list of travel solutions and the fare Quote BOM tree are constructed, the main function of the module can be called:

```
void quotePrices (const stdair::BookingRequestStruct&,
    stdair::TravelSolutionList_T&);
```

For each travel solution of the list, the applicable fare rules are picked from the BOM tree (information such as the trip type or the booking request date are only contained into the booking request, that is why we need this object too).

Each chosen fare rule enables to create a fare option structure which is finally stored into the travel solution.

12.7 Error Messages

This section lists the fatal errors you may encounter when using SimFQT:

- [Fare input file not found](#)
- [Fare input file can not be parsed](#)
- [Error Messages for missing fare rules](#)

12.7.1 Fare input file not found

In this case, the output error message will be similar to:

```
terminate called after throwing an instance of 'SIMFQT::FareInputFileNotFoundException'
  what():  The fare input file '~/<YourFileName>.csv' does not exist or can not be read
Aborted
```

You can check:

- the given path to your input file is correct.
- the specified file name <YourFileName> is correct.
- the file permission settings: is the file "readable"?

12.7.2 Fare input file can not be parsed

This error message means that your input file has been opened but has not been fully read.

```
terminate called after throwing an instance of 'SIMFQT::FareFileParsingFailedException'
  what():  Parsing of fare input file: ~/<YourFileName>.csv failed
Aborted
```

Your input file structure is somehow incorrect. See the tutorial section [How to build a fare input file?](#).

12.7.3 Error Messages for missing fare rules

If you obtain one of the error messages below and you are currently using your own input file, that means it has been fully read. However, at least one fare rule is missing to complete the fare quote.

- If your error message is about a missing airport pair, you should obtain a similar report:

```
terminate called after throwing an instance of 'SIMFQT::AirportPairNotFoundException'
  what():  No available fare rule for the Origin-Destination pair: xxx, xxx
Aborted
```

You need to be sure that all your travel solutions have at least one corresponding origin-destination fare rule. It seems you should add one origin-destination (i.e., xxx, xxx) fare rule into your input file.

- If your error message is about a missing fare rule for a flight date, you should obtain a similar report:

```
terminate called after throwing an instance of 'SIMFQT::FlightDateNotFoundException'
  what():  No available fare rule for the flight date x, xxxx-xxx-xx and to the Origin-Destination pair:
Aborted
```

You need to be sure that all your travel solutions have at least one corresponding fare rule: same origin-destination and valid date range. It seems you should add/change a fare rule with the Origin-Destination pair: xxx, xxx: its date range must include the flight date xxxx-xxx-xx.

- If your error message is about a missing fare rule for a point-of sale and/or channel, you should obtain a similar report:

```
terminate called after throwing an instance of 'SIMFQT::PosOrChannelNotFoundException'
  what():  No available fare rule for the point of sale xxx, the channel xx, the flight date x, xxxx-xxx-xx
Aborted
```

You need to be sure that all your travel solutions have at least one corresponding fare rule: same origin-destination, valid date range, same point-of-sale and same channel. It seems you should add/change a fare rule to have the same combination as given in the output error message: "the point of sale xxx, the channel xx, the flight date x, xxxx-xxx-xx and the Origin-Destination pair: xxx, xxx".

- If your error message is about a missing fare rule for a flight time, you should obtain a similar report:

```
terminate called after throwing an instance of 'SIMFQT::FlightTimeNotFoundException'
  what(): No available fare rule corresponding to 'xx; x, xxxx-xxx-xx; xxx, xxx; xx:xx' (parsed key) and
Aborted
```

You need to be sure that all your travel solutions have at least one corresponding fare rule: same origin-destination, valid date range, same point-of-sale, same channel and valid time range. Add/change a fare rule if necessary.

- If your error message is about a missing fare rule for some features, you should obtain a similar report:

```
terminate called after throwing an instance of 'SIMFQT::FeaturesNotFoundException'
  what(): No available fare rule corresponding to a trip type xx, to a stay duration of x, to a request
Aborted
```

You need to be sure that all your travel solutions have at least one corresponding fare rule: same origin-destination, valid date range, same point-of-sale, same channel, valid time range and valid features. The features are:

- the trip type. Maybe you need both "OW" (One-Way) and "RT" (Round-trip) fare rules?
- the minimum stay duration. You can try "0" for this parameter to include all the possible stay durations.
- the advance purchase. You can try "0" for this parameter to include all the booking requests up to departure date.

- If your error message is about a missing fare rule for an airline, you should obtain a similar report:

```
terminate called after throwing an instance of 'SIMFQT::AirlineNotFoundException'
  what(): No available fare rule corresponding to 'xx; x, xxxx-xxx-xx; xxx, xxx; xx:xx' (parsed key), to
Aborted
```

At least one of your fare rules is correct except that the fare into question must be defined by the airline operating (see the first two letters of the parsed key in the error message to know which airline is operating).

13 Supported Systems

13.1 Table of Contents

- [Introduction](#)
- [SimFQT 3.10.x](#)
 - [Linux Systems](#)
 - * [Fedora Core 4 with ATLAS](#)
 - * [Gentoo Linux with ACML](#)
 - * [Gentoo Linux with ATLAS](#)
 - * [Gentoo Linux with MKL](#)
 - * [Gentoo Linux with NetLib's BLAS and LAPACK](#)
 - * [Red Hat Enterprise Linux with SimFQT External](#)
 - * [SUSE Linux 10.0 with NetLib's BLAS and LAPACK](#)
 - * [SUSE Linux 10.0 with MKL](#)
 - [Windows Systems](#)
 - * [Microsoft Windows XP with Cygwin](#)
 - * [Microsoft Windows XP with Cygwin and ATLAS](#)
 - * [Microsoft Windows XP with Cygwin and ACML](#)
 - * [Microsoft Windows XP with MinGW, MSYS and ACML](#)

- * [Microsoft Windows XP with MinGW, MSYS and SimFQT External](#)
 - * [Microsoft Windows XP with MS Visual C++ and Intel MKL](#)
- [Unix Systems](#)
 - * [SunOS 5.9 with SimFQT External](#)
- [SimFQT 3.9.1](#)
- [SimFQT 3.9.0](#)
- [SimFQT 3.8.1](#)

13.2 Introduction

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

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

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

13.3 SimFQT 3.10.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
- **SimFQT release:** 3.10.0
- **External Libraries:** From FC4 distribution:
 - `fftw3.i386-3.0.1-3`
 - `fftw3-devel.i386-3.0.1-3`
 - `atlas-sse2.i386-3.6.0-8.fc4`
 - `atlas-sse2-devel.i386-3.6.0-8.fc4`
 - `blas.i386-3.0-35.fc4`
 - `lapack.i386-3.0-35.fc4`
- **Tests Status:** All tests PASSED
- **Comments:** SimFQT 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
- **SimFQT release:** 3.10.1
- **External Libraries:** Compiled and installed from portage tree:
 - sci-libs/acml-3.0.0
- **Tests Status:** All tests PASSED
- **Comments:** BLAS and LAPACK libs set by using the following system commands:

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

SimFQT 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
- **SimFQT release:** 3.10.1
- **External Libraries:** Compiled and installed from portage tree:
 - sci-libs/fftw-3.1
 - sci-libs/blas-atlas-3.6.0-r1
 - sci-libs/lapack-atlas-3.6.0
- **Tests Status:** All tests PASSED
- **Comments:** BLAS and LAPACK libs set by using the following system commands:

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

SimFQT 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
- **SimFQT release:** 3.10.0
- **External Libraries:** Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory:
/opt/intel/mkl/8.0.1
- **Tests Status:** All tests PASSED
- **Comments:** SimFQT 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
- **SimFQT release:** 3.10.1
- **External Libraries:** Compiled and installed from portage tree:
 - sci-libs/fftw-3.1
 - sci-libs/blas-reference-19940131-r2
 - sci-libs/cblas-reference-20030223
 - sci-libs/lapack-reference-3.0-r2
- **Tests Status:** All tests PASSED
- **Comments:** BLAS and LAPACK libs set by using the following system commands:

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

SimFQT configured with:

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

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

13.3.1.6 Red Hat Enterprise Linux with SimFQT 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)
- **SimFQT release:** 3.10.0
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from SimFQT 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
- **SimFQT release:** 3.10.0
- **External Libraries:** BLAS, LAPACK and FFTW libraries installed from OpenSuse 10.0 RPM repository:
 - blas-3.0-926
 - lapack-3.0-926
 - fftw3-3.0.1-114
 - fftw3-threads-3.0.1-114
 - fftw3-devel-3.0.1-114
- **Tests Status:** All tests PASSED
- **Comments:** SimFQT 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
- **SimFQT release:** 3.10.0
- **External Libraries:** Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory:


```
/opt/intel/mkl/8.0.1
```
- **Tests Status:** All tests PASSED
- **Comments:** SimFQT 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)
- **SimFQT release:** 3.10.1
- **External Libraries:** Installed from Cygwin's repository:
 - fftw-3.0.1-2
 - fftw-dev-3.0.1-1
 - lapack-3.0-4
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. SimFQT 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)
- **SimFQT release:** 3.10.1
- **External Libraries:** Installed from Cygwin's repository:
 - fftw-3.0.1-2
 - fftw-dev-3.0.1-1ATLAS BLAS and LAPACK libraries from SimFQT External 2.1.1 package configured using:

```
% ./configure --enable-atlas --disable-fftw
```
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. SimFQT 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)
- **SimFQT release:** 3.10.2
- **External Libraries:** ACML version 3.1.0 (acml3.1.0-32-win32-g77.exe) installed into a default directory, i.e. "c:\Program Files\AMD\acml3.1.0"
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. SimFQT 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)
- **SimFQT release:** 3.10.2
- **External Libraries:** ACML version 3.1.0 (acml3.1.0-32-win32-g77.exe) installed into a default directory, i.e. "c:\Program Files\AMD\acml3.1.0"
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. SimFQT 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 SimFQT 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)
- **SimFQT release:** 3.10.5
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from SimFQT External 2.2.0 package
- **Tests Status:** All tests PASSED
- **Comments:** Only static library can be built. SimFQT 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
- **SimFQT release:** 3.10.5
- **External Libraries:** Intel Math Kernel Library (MKL) 8.1 installed manually in the following directory: "C:\Program Files\Intel\MKL\8.1"
- **Tests Status:** Not fully tested. Some SimFQT based programs compiled and run with success.
- **Comments:** Only static library can be built. SimFQT built by opening the "win32\simfqt.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 SimFQT External

- **Platform:** SUNW, Sun-Blade-100 (SPARC)
- **Operating System:** SunOS 5.9 Generic_112233-10
- **Compiler(s):** g++ (GCC) 3.4.5
- **SimFQT release:** 3.10.2
- **External Libraries:** BLAS, CBLAS, LAPACK and FFTW libraries from SimFQT 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:** SimFQT 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 SimFQT Supported Systems (Previous Releases)

14.1 SimFQT 3.9.1

14.2 SimFQT 3.9.0

14.3 SimFQT 3.8.1

15 Tutorials

15.1 Table of Contents

- [Preparing the SimFQT Project for Development](#)
- [Your first fareQuote](#)
 - [Summary of the different steps](#)
 - [Result of the Batch Program](#)
- [Fare quoting with an input file](#)
 - [How to build a fare input file?](#)
 - [Building the BOM tree with an input file](#)
 - [Result of the Batch Program](#)

15.2 Preparing the SimFQT Project for Development

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

15.3 Your first fareQuote

15.3.1 Summary of the different steps

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

First, we instantiate the `simfqtService` object:

```
std::ofstream logOutputFile;
const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
SIMFQT::SIMFQT_Service simfqtService (lLogParams);
```

Then, we construct a default sample list of travel solutions and a default booking request (as mentionned in [Instantiate the default booking request](#) and [Instantiate the default travel solution list](#) parts):

```
stdair::TravelSolutionList_T& ioInteractiveTravelSolutionList,
return ioBookingRequestStruct;
```

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

```
simfqtService.buildSampleBom();
```

The main step is the fare quoting (see [The fare quoting procedure](#)):

```
simfqtService.quotePrices (lInteractiveBookingRequest,
```

15.3.2 Result of the Batch Program

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

```
[D]../../../../simfqt/batches/simfqt.cpp:186: Welcome to Simfqt
[D]../../../../simfqt/batches/simfqt.cpp:212: Travel solutions:
[0] [0] BA, 9, 2011-06-10, LHR, SYD, 21:45 --- ---
[D]../../../../simfqt/command/FareQuoter.cpp:519: Segment path: BA; 9, 2011-06-10;
LHR, SYD; 21:45. A corresponding fare option for the 'BA Y' class is: Class
path: Y; 450 EUR; conditions: 1 1 1
[D]../../../../simfqt/service/SIMFQT_Service.cpp:352: Fare Quote retrieving: 0.001
403 - SIMFQT_ServiceContext -- Owns StdAir service: 1
[D]../../../../simfqt/batches/simfqt.cpp:214: BOM tree:
=====
BomRoot: -- ROOT --
=====
+++++
AirportPair: LHR, SYD
+++++
-----
DatePeriod: [2011-Jan-15/2011-Dec-30]
-----
*****
PosChannel: LHR,DN
*****
-----
TimePeriod: 00:00:00-23:00:00
-----
-----
Fare-Features: RT -- 0-1-1-0
-----
-----
AirlineClassList: BA Y
-----

[D]../../../../simfqt/batches/simfqt.cpp:219: Travel solutions:
[0] [0] BA, 9, 2011-06-10, LHR, SYD, 21:45 --- Y, 450, 1 1 1 ---
```

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

```
[0] [0] BA, 9, 2011-06-10, LHR, SYD, 21:45 --- ---
```

and after the fare quoting:

```
[0] [0] BA, 9, 2011-06-10, LHR, SYD, 21:45 --- Y, 450, 1 1 1 ---
```

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

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

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

15.4 Fare quoting with an input file

15.4.1 How to build a fare input file?

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

```
stdair::TravelSolutionList_T& ioInteractiveTravelSolutionList,
```

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

```
[0] [0] BA, 9, 2011-06-10, LHR, SYD, 21:45 --- ---
```

We deduce:

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

We can deduce a part of our fare rule file :

```
// Fares: fare ID; OriginCity; DestinationCity; TripType; DateRangeStart;
    DateRangeEnd; DepartureTimeRangeStart; DepartureTimeRangeEnd; POS; CabinCode;
    Channel; AdvancePurchase; SaturdayNight; ChangeFees; NonRefundable; MinimumStay; Price;
    nb Segments
// Segment: AirlineCode; Class;
1; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; ???; ?; ??; ?; ?; ?; ?;
    ?; ???; BA; ?;
```

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

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

The input file is now:

```
// Fares: fare ID; OriginCity; DestinationCity; TripType; DateRangeStart;
    DateRangeEnd; DepartureTimeRangeStart; DepartureTimeRangeEnd; POS; CabinCode;
    Channel; AdvancePurchase; SaturdayNight; ChangeFees; NonRefundable; MinimumStay; Price;
    nb Segments
// Segment: AirlineCode; Class;
1; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; ?; IN; 0; ?; ?; ?;
    0; ???; BA; ?;
2; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; ?; IF; 0; ?; ?; ?;
    0; ???; BA; ?;
3; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; ?; DN; 0; ?; ?; ?;
    0; ???; BA; ?;
4; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; ?; DF; 0; ?; ?; ?;
    0; ???; BA; ?;
5; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; ?; IN; 0; ?; ?; ?;
    0; ???; BA; ?;
6; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; ?; IF; 0; ?; ?; ?;
    0; ???; BA; ?;
7; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; ?; DN; 0; ?; ?; ?;
    0; ???; BA; ?;
8; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; ?; DF; 0; ?; ?; ?;
    0; ???; BA; ?;
9; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; ?; IN; 0; ?; ?; ?;
    0; ???; BA; ?;
10; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; ?; IF; 0; ?; ?; ?;
    0; ???; BA; ?;
11; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; ?; DN; 0; ?; ?; ?;
    0; ???; BA; ?;
12; LHR; SYD; ??; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; ?; DF; 0; ?; ?; ?;
    0; ???; BA; ?;
```

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

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

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

We obtain:

```
// Fares: fare ID; OriginCity; DestinationCity; TripType; DateRangeStart;
    DateRangeEnd; DepartureTimeRangeStart; DepartureTimeRangeEnd; POS; CabinCode;
    Channel; AdvancePurchase; SaturdayNight; ChangeFees; NonRefundable; MinimumStay; Price;
    nb Segments
// Segment: AirlineCode; Class;
1; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; IN; 0; T; T; T;
    0; 50; BA; Y;
```

```

2; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; IF; 0; T; T; T;
0; 150; BA; Y;
3; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; DN; 0; T; T; T;
0; 250; BA; Y;
4; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; DF; 0; T; T; T;
0; 350; BA; Y;
5; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; IN; 0; T; T; T;
0; 450; BA; Y;
6; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; IF; 0; T; T; T;
0; 550; BA; Y;
7; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; DN; 0; T; T; T;
0; 650; BA; Y;
8; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; DF; 0; T; T; T;
0; 750; BA; Y;
9; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; IN; 0; T; T; T;
0; 850; BA; Y;
10; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; IF; 0; T; T; T;
0; 950; BA; Y;
11; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; DN; 0; T; T; T;
0; 1050; BA; Y;
12; LHR; SYD; OW; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; DF; 0; T; T; T;
0; 1150; BA; Y;
13; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; IN; 0; T; T; T;
0; 90; BA; Y;
14; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; IF; 0; T; T; T;
0; 190; BA; Y;
15; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; DN; 0; T; T; T;
0; 290; BA; Y;
16; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; LHR; Y; DF; 0; T; T; T;
0; 390; BA; Y;
17; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; IN; 0; T; T; T;
0; 490; BA; Y;
18; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; IF; 0; T; T; T;
0; 590; BA; Y;
19; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; DN; 0; T; T; T;
0; 690; BA; Y;
20; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; SYD; Y; DF; 0; T; T; T;
0; 790; BA; Y;
21; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; IN; 0; T; T; T;
0; 890; BA; Y;
22; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; IF; 0; T; T; T;
0; 990; BA; Y;
23; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; DN; 0; T; T; T;
0; 1090; BA; Y;
24; LHR; SYD; RT; 2011-01-01; 2011-12-31; 00:00; 23:59; ROW; Y; DF; 0; T; T; T;
0; 1190; BA; Y;

```

15.4.2 Building the BOM tree with an input file

The steps are the same as before [Summary of the different steps](#) except the bom tree must be built using the fare input file :

15.4.3 Result of the Batch Program

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

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

the last lines of the log output should look like:

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

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

16 Command-Line Test to Demonstrate How To Test the SimFQT 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 FQTTestSuite
#include <boost/test/unit_test.hpp>
// StdAir
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
// SimFQT
#include <simfqt/SIMFQT_Service.hpp>
#include <simfqt/config/simfqt-paths.hpp>

namespace boost_utf = boost::unit_test;

struct UnitTestConfig {
    UnitTestConfig() {
        static std::ofstream _test_log ("FQTTestSuite_utfresults.xml");
        boost_utf::unit_test_log.set_stream (_test_log);
        boost_utf::unit_test_log.set_format (boost_utf::XML);
        boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
        //boost_utf::unit_test_log.set_threshold_level
            (boost_utf::log_successful_tests);
    }

    ~UnitTestConfig() {
    }
};

// ////////////////////////////////////////
void testFareQuoterHelper (const unsigned short iTestFlag,
                          const stdair::Filename_T iFareInputFilename,
                          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 SimFQT service object
    const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
                                           logOutputFile);

    // Initialise the Simfqt service object
    SIMFQT::SIMFQT_Service simfqtService (lLogParams);

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

        // Build the default sample BOM tree (filled with fares) for Simfqt
        simfqtService.buildSampleBom();
    } else {

        // Build the BOM tree from parsing the fare input file
        SIMFQT::FareFilePath lFareFilePath (iFareInputFilename)
        ;
        simfqtService.parseAndLoad (lFareFilePath);
    }

    // Build a sample list of travel solutions and a booking request.
    stdair::TravelSolutionList_T lTravelSolutionList;
    simfqtService.buildSampleTravelSolutions (lTravelSolutionList);
    stdair::BookingRequestStruct lBookingRequest =
        simfqtService.buildBookingRequest();

    // Try to fareQuote the sample list of travel solutions
    simfqtService.quotePrices (lBookingRequest, 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 (UnitTestFixture);

// Start the test suite
BOOST_AUTO_TEST_SUITE (master_test_suite)

BOOST_AUTO_TEST_CASE (simfqt_simple_pricing_test) {

    // Input file name
    const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
        "/fare01.csv");

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

    // Try to fareQuote the sample default list of travel solutions
    BOOST_CHECK_NO_THROW (testFareQuoterHelper (0, lFareInputFilename, isBuiltin)
        );
}

BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_01) {

    // Input file name
    const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
        "/fareError01.csv");

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

    // Try to fareQuote the sample default list of travel solutions
    BOOST_CHECK_THROW (testFareQuoterHelper (1, lFareInputFilename, isBuiltin),
        SIMFQT::AirportPairNotFoundException
    );
}

BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_02) {

    // Input file name
    const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
        "/fareError02.csv");

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

    // Try to fareQuote the sample default list of travel solutions
    BOOST_CHECK_THROW (testFareQuoterHelper (2, lFareInputFilename, isBuiltin),
        SIMFQT::PosOrChannelNotFoundException
    );
}

BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_03) {

    // Input file name
    const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
        "/fareError03.csv");

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

    // Try to fareQuote the sample default list of travel solutions
    BOOST_CHECK_THROW (testFareQuoterHelper (3, lFareInputFilename, isBuiltin),
        SIMFQT::FlightDateNotFoundException
    );
}

BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_04) {

    // Input file name
    const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
        "/fareError04.csv");

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

    // Try to fareQuote the sample default list of travel solutions
    BOOST_CHECK_THROW (testFareQuoterHelper (4, lFareInputFilename, isBuiltin),
        SIMFQT::FlightTimeNotFoundException
    );
}
```



```

BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_05) {

    // Input file name
    const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
        "/fareError05.csv");

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

    // Try to fareQuote the sample default list of travel solutions
    BOOST_CHECK_THROW (testFareQuoterHelper (5, lFareInputFilename, isBuiltin),
        SIMFQT::FeaturesNotFoundException
    );
}

BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_06) {

    // Input file name
    const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
        "/fareError06.csv");

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

    // Try to fareQuote the sample default list of travel solutions
    BOOST_CHECK_THROW (testFareQuoterHelper (6, lFareInputFilename, isBuiltin),
        SIMFQT::AirlineNotFoundException
    );
}

BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_07) {

    // Input file name
    const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
        "/fareError07.csv");

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

    // Try to fareQuote the sample default list of travel solutions
    BOOST_CHECK_THROW (testFareQuoterHelper (7, lFareInputFilename, isBuiltin),
        SIMFQT::FareFileParsingFailedException
    );
}

BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_08) {

    // Input file name
    const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
        "/missingFile.csv");

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

    // Try to fareQuote the sample default list of travel solutions
    BOOST_CHECK_THROW (testFareQuoterHelper (8, lFareInputFilename, isBuiltin),
        SIMFQT::FareInputFileNotFoundException
    );
}

BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_09) {

    // Input file name
    const stdair::Filename_T lEmptyInputFilename (STDAIR_SAMPLE_DIR
        "/ ");

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

    // Try to fareQuote the sample default list of travel solutions
    BOOST_CHECK_NO_THROW (testFareQuoterHelper (9, lEmptyInputFilename, isBuiltin)
    );
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

/*!

```

17 Namespace Index

17.1 Namespace List

Here is a list of all namespaces with brief descriptions:

SIMFQT	53
SIMFQT::FareParserHelper	54
stdair	
Forward declarations	55

18 Class Index

18.1 Class Hierarchy

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

std::basic_fstream< char >	
std::basic_fstream< wchar_t >	
std::basic_ifstream< char >	
std::basic_ifstream< wchar_t >	
std::basic_ios< char >	
std::basic_ios< wchar_t >	
std::basic_iostream< char >	
std::basic_iostream< wchar_t >	
std::basic_istream< char >	
std::basic_istream< wchar_t >	
std::basic_istreamstream< char >	
std::basic_istreamstream< wchar_t >	
std::basic_ofstream< char >	
std::basic_ofstream< wchar_t >	
std::basic_ostream< char >	
std::basic_ostream< wchar_t >	
std::basic_ostreamstream< char >	
std::basic_ostreamstream< wchar_t >	
std::basic_string< char >	
std::basic_string< wchar_t >	
std::basic_stringstream< char >	
std::basic_stringstream< wchar_t >	
CmdAbstract	57
SIMFQT::FareParser	62
SIMFQT::FareRuleFileParser	63
SIMFQT::FareRuleGenerator	64
FacServiceAbstract	59
SIMFQT::FacSimfqtServiceContext	59
SIMFQT::FareQuoter	63
FileNotFoundException	79
SIMFQT::FareInputFileNotFoundException	62
grammar	80

SIMFQT::FareParserHelper::FareRuleParser< Iterator >	65
InputFilePath	81
SIMFQT::FareFilePath	61
ObjectNotFoundException	81
SIMFQT::AirlineNotFoundException	56
SIMFQT::AirportPairNotFoundException	56
SIMFQT::FeaturesNotFoundException	78
SIMFQT::FlightDateNotFoundException	79
SIMFQT::FlightTimeNotFoundException	80
SIMFQT::PosOrChannelNotFoundException	83
SIMFQT::FareParserHelper::ParserSemanticAction	81
SIMFQT::FareParserHelper::doEndFare	57
SIMFQT::FareParserHelper::storeAdvancePurchase	90
SIMFQT::FareParserHelper::storeAirlineCode	91
SIMFQT::FareParserHelper::storeCabinCode	92
SIMFQT::FareParserHelper::storeChangeFees	93
SIMFQT::FareParserHelper::storeChannel	95
SIMFQT::FareParserHelper::storeClass	96
SIMFQT::FareParserHelper::storeDateRangeEnd	97
SIMFQT::FareParserHelper::storeDateRangeStart	98
SIMFQT::FareParserHelper::storeDestination	100
SIMFQT::FareParserHelper::storeEndRangeTime	101
SIMFQT::FareParserHelper::storeFare	102
SIMFQT::FareParserHelper::storeFareId	104
SIMFQT::FareParserHelper::storeMinimumStay	105
SIMFQT::FareParserHelper::storeNonRefundable	106
SIMFQT::FareParserHelper::storeOrigin	107
SIMFQT::FareParserHelper::storePOS	109
SIMFQT::FareParserHelper::storeSaturdayStay	110
SIMFQT::FareParserHelper::storeStartRangeTime	111
SIMFQT::FareParserHelper::storeTripType	112
ParsingFileFailedException	83

SIMFQT::FareFileParsingFailedException	60
RootException	84
SIMFQT::QuotingException	84
ServiceAbstract	85
SIMFQT::SIMFQT_ServiceContext	89
SIMFQT::SIMFQT_Service	85
StructAbstract	114
SIMFQT::FareRuleStruct	71

19 Class Index

19.1 Class List

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

SIMFQT::AirlineNotFoundException	56
SIMFQT::AirportPairNotFoundException	56
CmdAbstract	57
SIMFQT::FareParserHelper::doEndFare	57
FacServiceAbstract	59
SIMFQT::FacSimfqtServiceContext Factory for the service context	59
SIMFQT::FareFileParsingFailedException	60
SIMFQT::FareFilePath	61
SIMFQT::FareInputFileNotFoundException	62
SIMFQT::FareParser	62
SIMFQT::FareQuoter Command wrapping the pricing request process	63
SIMFQT::FareRuleFileParser	63
SIMFQT::FareRuleGenerator	64
SIMFQT::FareParserHelper::FareRuleParser< Iterator >	65
SIMFQT::FareRuleStruct	71
SIMFQT::FeaturesNotFoundException	78
FileNotFoundException	79
SIMFQT::FlightDateNotFoundException	79

SIMFQT::FlightTimeNotFoundException	80
grammar	80
InputFilePath	81
ObjectNotFoundException	81
SIMFQT::FareParserHelper::ParserSemanticAction	81
ParsingFileFailedException	83
SIMFQT::PosOrChannelNotFoundException	83
SIMFQT::QuotingException	84
RootException	84
ServiceAbstract	85
SIMFQT::SIMFQT_Service Interface for the SIMFQT Services	85
SIMFQT::SIMFQT_ServiceContext Class holding the context of the SimFQT services	89
SIMFQT::FareParserHelper::storeAdvancePurchase	90
SIMFQT::FareParserHelper::storeAirlineCode	91
SIMFQT::FareParserHelper::storeCabinCode	92
SIMFQT::FareParserHelper::storeChangeFees	93
SIMFQT::FareParserHelper::storeChannel	95
SIMFQT::FareParserHelper::storeClass	96
SIMFQT::FareParserHelper::storeDateRangeEnd	97
SIMFQT::FareParserHelper::storeDateRangeStart	98
SIMFQT::FareParserHelper::storeDestination	100
SIMFQT::FareParserHelper::storeEndRangeTime	101
SIMFQT::FareParserHelper::storeFare	102
SIMFQT::FareParserHelper::storeFareId	104
SIMFQT::FareParserHelper::storeMinimumStay	105
SIMFQT::FareParserHelper::storeNonRefundable	106
SIMFQT::FareParserHelper::storeOrigin	107
SIMFQT::FareParserHelper::storePOS	109
SIMFQT::FareParserHelper::storeSaturdayStay	110
SIMFQT::FareParserHelper::storeStartRangeTime	111
SIMFQT::FareParserHelper::storeTripType	112

StructAbstract	114
--------------------------------	-----

20 File Index

20.1 File List

Here is a list of all files with brief descriptions:

simfqt/SIMFQT_Service.hpp	167
simfqt/SIMFQT_Types.hpp	169
simfqt/basic/BasConst.cpp	115
simfqt/basic/BasConst_General.hpp	115
simfqt/basic/BasConst_SIMFQT_Service.hpp	116
simfqt/batches/simfqt_parseFareRules.cpp	118
simfqt/bom/FareRuleStruct.cpp	121
simfqt/bom/FareRuleStruct.hpp	122
simfqt/command/FareParser.cpp	126
simfqt/command/FareParser.hpp	127
simfqt/command/FareParserHelper.cpp	128
simfqt/command/FareParserHelper.hpp	138
simfqt/command/FareQuoter.cpp	141
simfqt/command/FareQuoter.hpp	149
simfqt/command/FareRuleGenerator.cpp	151
simfqt/command/FareRuleGenerator.hpp	154
simfqt/config/simfqt-paths.hpp	157
simfqt/factory/FacSimfqtServiceContext.cpp	157
simfqt/factory/FacSimfqtServiceContext.hpp	158
simfqt/service/SIMFQT_Service.cpp	159
simfqt/service/SIMFQT_ServiceContext.cpp	164
simfqt/service/SIMFQT_ServiceContext.hpp	165
simfqt/ui/cmdline/simfqt.cpp	170
test/simfqt/FQTTestSuite.cpp	184

21 Namespace Documentation

21.1 SIMFQT Namespace Reference

Namespaces

- namespace [FareParserHelper](#)

Classes

- struct [FareRuleStruct](#)
- class [FareParser](#)
- class [FareRuleFileParser](#)
- class [FareQuoter](#)
 - Command wrapping the pricing request process.*
- class [FareRuleGenerator](#)
- class [FacSimfqtServiceContext](#)
 - Factory for the service context.*
- class [SIMFQT_ServiceContext](#)
 - Class holding the context of the SimFQT services.*
- class [SIMFQT_Service](#)
 - Interface for the SIMFQT Services.*
- class [FareFileParsingFailedException](#)
- class [AirportPairNotFoundException](#)
- class [PosOrChannelNotFoundException](#)
- class [FlightDateNotFoundException](#)
- class [FlightTimeNotFoundException](#)
- class [FeaturesNotFoundException](#)
- class [AirlineNotFoundException](#)
- class [FareInputFileNotFoundException](#)
- class [QuotingException](#)
- class [FareFilePath](#)

Typedefs

- typedef unsigned int [FareQuoteID_T](#)
- typedef boost::shared_ptr
 - < [SIMFQT_Service](#) > [SIMFQT_ServicePtr_T](#)

Variables

- const std::string [DEFAULT_FARE_QUOTER_ID](#) = "IATA"

21.1.1 Typedef Documentation

21.1.1.1 typedef unsigned int SIMFQT::FareQuoteID_T

ID for the Fare Quote system.

Definition at line 143 of file [SIMFQT_Types.hpp](#).

21.1.1.2 typedef boost::shared_ptr<SIMFQT_Service> SIMFQT::SIMFQT_ServicePtr_T

(Smart) Pointer on the SimFQT service handler.

Definition at line 148 of file [SIMFQT_Types.hpp](#).

21.1.2 Variable Documentation

21.1.2.1 `const std::string SIMFQT::DEFAULT_FARE_QUOTER_ID = "IATA"`

Default ID for the [SIMFQT_Service](#).

Definition at line 10 of file [BasConst.cpp](#).

21.2 SIMFQT::FareParserHelper Namespace Reference

Classes

- struct [FareRuleParser](#)
- struct [ParserSemanticAction](#)
- struct [storeFareId](#)
- struct [storeOrigin](#)
- struct [storeDestination](#)
- struct [storeTripType](#)
- struct [storeDateRangeStart](#)
- struct [storeDateRangeEnd](#)
- struct [storeStartRangeTime](#)
- struct [storeEndRangeTime](#)
- struct [storePOS](#)
- struct [storeCabinCode](#)
- struct [storeChannel](#)
- struct [storeAdvancePurchase](#)
- struct [storeSaturdayStay](#)
- struct [storeChangeFees](#)
- struct [storeNonRefundable](#)
- struct [storeMinimumStay](#)
- struct [storeFare](#)
- struct [storeAirlineCode](#)
- struct [storeClass](#)
- struct [doEndFare](#)

Variables

- `stdair::int1_p_t` [int1_p](#)
- `stdair::uint2_p_t` [uint2_p](#)
- `stdair::uint4_p_t` [uint4_p](#)
- `stdair::uint1_4_p_t` [uint1_4_p](#)
- `stdair::hour_p_t` [hour_p](#)
- `stdair::minute_p_t` [minute_p](#)
- `stdair::second_p_t` [second_p](#)
- `stdair::year_p_t` [year_p](#)
- `stdair::month_p_t` [month_p](#)
- `stdair::day_p_t` [day_p](#)

21.2.1 Variable Documentation

21.2.1.1 `stdair::int1_p_t SIMFQT::FareParserHelper::int1_p`

Namespaces. 1-digit-integer parser

Definition at line 444 of file [FareParserHelper.cpp](#).

21.2.1.2 stdair::uint2_p_t SIMFQT::FareParserHelper::uint2_p

2-digit-integer parser

Definition at line 447 of file [FareParserHelper.cpp](#).

21.2.1.3 stdair::uint4_p_t SIMFQT::FareParserHelper::uint4_p

4-digit-integer parser

Definition at line 450 of file [FareParserHelper.cpp](#).

21.2.1.4 stdair::uint1_4_p_t SIMFQT::FareParserHelper::uint1_4_p

Up-to-4-digit-integer parser

Definition at line 453 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

21.2.1.5 stdair::hour_p_t SIMFQT::FareParserHelper::hour_p

Time element parsers.

Definition at line 456 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

21.2.1.6 stdair::minute_p_t SIMFQT::FareParserHelper::minute_p

Definition at line 457 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

21.2.1.7 stdair::second_p_t SIMFQT::FareParserHelper::second_p

Definition at line 458 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

21.2.1.8 stdair::year_p_t SIMFQT::FareParserHelper::year_p

Date element parsers.

Definition at line 461 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

21.2.1.9 stdair::month_p_t SIMFQT::FareParserHelper::month_p

Definition at line 462 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

21.2.1.10 stdair::day_p_t SIMFQT::FareParserHelper::day_p

Definition at line 463 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

21.3 stdair Namespace Reference

Forward declarations.

21.3.1 Detailed Description

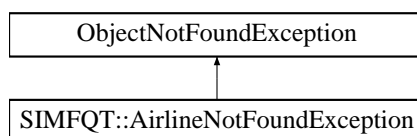
Forward declarations.

22 Class Documentation

22.1 SIMFQT::AirlineNotFoundException Class Reference

```
#include <simfqt/SIMFQT_Types.hpp>
```

Inheritance diagram for SIMFQT::AirlineNotFoundException:



Public Member Functions

- [AirlineNotFoundException](#) (const std::string &iWhat)

22.1.1 Detailed Description

The airline can not be found.

Definition at line 99 of file [SIMFQT_Types.hpp](#).

22.1.2 Constructor & Destructor Documentation

22.1.2.1 `SIMFQT::AirlineNotFoundException::AirlineNotFoundException (const std::string & iWhat) [inline]`

Constructor.

Definition at line 104 of file [SIMFQT_Types.hpp](#).

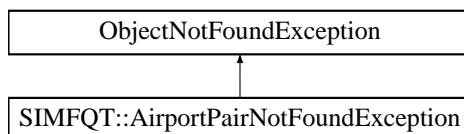
The documentation for this class was generated from the following file:

- [simfqt/SIMFQT_Types.hpp](#)

22.2 SIMFQT::AirportPairNotFoundException Class Reference

```
#include <simfqt/SIMFQT_Types.hpp>
```

Inheritance diagram for SIMFQT::AirportPairNotFoundException:



Public Member Functions

- [AirportPairNotFoundException](#) (const std::string &iWhat)

22.2.1 Detailed Description

The given airport pair can not be found.

Definition at line 39 of file [SIMFQT_Types.hpp](#).

22.2.2 Constructor & Destructor Documentation

22.2.2.1 SIMFQT::AirportPairNotFoundException::AirportPairNotFoundException (const std::string & iWhat) [inline]

Constructor.

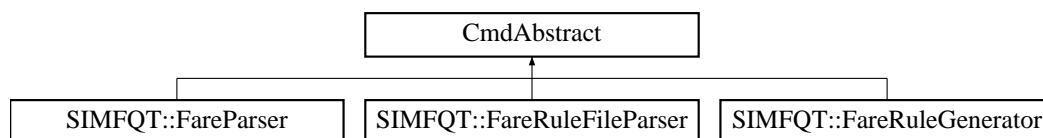
Definition at line 44 of file [SIMFQT_Types.hpp](#).

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

- [simfqt/SIMFQT_Types.hpp](#)

22.3 CmdAbstract Class Reference

Inheritance diagram for CmdAbstract:



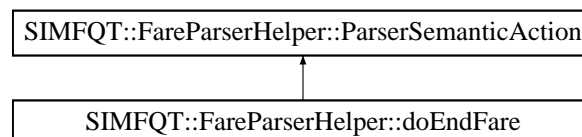
The documentation for this class was generated from the following file:

- [simfqt/command/FareRuleGenerator.hpp](#)

22.4 SIMFQT::FareParserHelper::doEndFare Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::doEndFare:



Public Member Functions

- [doEndFare](#) (stdair::BomRoot &, [FareRuleStruct](#) &)
- void [operator\(\)](#) (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [stdair::BomRoot](#) & [_bomRoot](#)
- [FareRuleStruct](#) & [_fareRule](#)

22.4.1 Detailed Description

Mark the end of the fare-rule parsing.

Definition at line 230 of file [FareParserHelper.hpp](#).

22.4.2 Constructor & Destructor Documentation

22.4.2.1 SIMFQT::FareParserHelper::doEndFare::doEndFare ([stdair::BomRoot](#) & [ioBomRoot](#), [FareRuleStruct](#) & [ioFareRule](#))

Actor Constructor.

Definition at line 417 of file [FareParserHelper.cpp](#).

22.4.3 Member Function Documentation

22.4.3.1 void SIMFQT::FareParserHelper::doEndFare::operator() ([boost::spirit::qi::unused_type](#) , [boost::spirit::qi::unused_type](#) , [boost::spirit::qi::unused_type](#)) const

Actor Function (functor).

Definition at line 424 of file [FareParserHelper.cpp](#).

References [_bomRoot](#), [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::describe\(\)](#).

22.4.4 Member Data Documentation

22.4.4.1 [stdair::BomRoot](#)& [SIMFQT::FareParserHelper::doEndFare::_bomRoot](#)

Actor Specific Context.

Definition at line 238 of file [FareParserHelper.hpp](#).

Referenced by [operator\(\)\(\)](#).

22.4.4.2 [FareRuleStruct](#)& [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#) [inherited]

Actor Context.

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

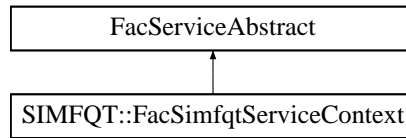
Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)\(\)](#), and [operator\(\)\(\)](#).

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

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.5 FacServiceAbstract Class Reference

Inheritance diagram for FacServiceAbstract:



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

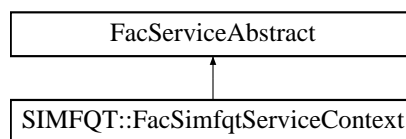
- [simfqt/factory/FacSimfqtServiceContext.hpp](#)

22.6 SIMFQT::FacSimfqtServiceContext Class Reference

Factory for the service context.

```
#include <simfqt/factory/FacSimfqtServiceContext.hpp>
```

Inheritance diagram for SIMFQT::FacSimfqtServiceContext:



Public Member Functions

- [~FacSimfqtServiceContext \(\)](#)
- [SIMFQT_ServiceContext & create \(\)](#)

Static Public Member Functions

- static [FacSimfqtServiceContext & instance \(\)](#)

Protected Member Functions

- [FacSimfqtServiceContext \(\)](#)

22.6.1 Detailed Description

Factory for the service context.

Definition at line 22 of file [FacSimfqtServiceContext.hpp](#).

22.6.2 Constructor & Destructor Documentation

22.6.2.1 SIMFQT::FacSimfqtServiceContext::~~FacSimfqtServiceContext ()

Destructor.

The Destruction put the `_instance` to NULL in order to be clean for the next [FacSimfqtServiceContext::instance\(\)](#).

Definition at line 17 of file [FacSimfqtServiceContext.cpp](#).

22.6.2.2 SIMFQT::FacSimfqtServiceContext::FacSimfqtServiceContext () [inline], [protected]

Default Constructor.

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

Definition at line 57 of file [FacSimfqtServiceContext.hpp](#).

Referenced by [instance\(\)](#).

22.6.3 Member Function Documentation

22.6.3.1 FacSimfqtServiceContext & SIMFQT::FacSimfqtServiceContext::instance () [static]

Provide the unique instance.

The singleton is instantiated when first used.

Returns

FacServiceContext&

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

References [FacSimfqtServiceContext\(\)](#).

22.6.3.2 SIMFQT_ServiceContext & SIMFQT::FacSimfqtServiceContext::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 [FacSimfqtServiceContext.cpp](#).

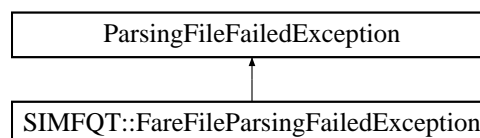
The documentation for this class was generated from the following files:

- [simfqt/factory/FacSimfqtServiceContext.hpp](#)
- [simfqt/factory/FacSimfqtServiceContext.cpp](#)

22.7 SIMFQT::FareFileParsingFailedException Class Reference

```
#include <simfqt/SIMFQT_Types.hpp>
```

Inheritance diagram for SIMFQT::FareFileParsingFailedException:



Public Member Functions

- [FareFileParsingFailedException](#) (const std::string &iWhat)

22.7.1 Detailed Description

The fare input file can not be parsed.

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

22.7.2 Constructor & Destructor Documentation

22.7.2.1 SIMFQT::FareFileParsingFailedException::FareFileParsingFailedException (const std::string & *iWhat*) [inline]

Constructor.

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

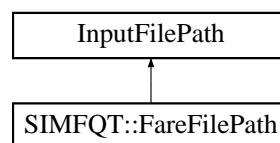
The documentation for this class was generated from the following file:

- [simfqt/SIMFQT_Types.hpp](#)

22.8 SIMFQT::FareFilePath Class Reference

```
#include <simfqt/SIMFQT_Types.hpp>
```

Inheritance diagram for SIMFQT::FareFilePath:



Public Member Functions

- [FareFilePath](#) (const stdair::Filename_T &iFilename)

22.8.1 Detailed Description

Fare input file.

Definition at line 130 of file [SIMFQT_Types.hpp](#).

22.8.2 Constructor & Destructor Documentation

22.8.2.1 SIMFQT::FareFilePath::FareFilePath (const stdair::Filename_T & *iFilename*) [inline], [explicit]

Constructor.

Definition at line 135 of file [SIMFQT_Types.hpp](#).

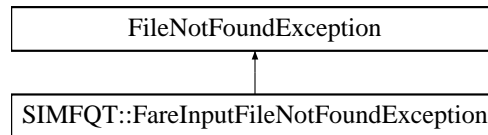
The documentation for this class was generated from the following file:

- [simfqt/SIMFQT_Types.hpp](#)

22.9 SIMFQT::FareInputFileNotFoundException Class Reference

```
#include <simfqt/SIMFQT_Types.hpp>
```

Inheritance diagram for SIMFQT::FareInputFileNotFoundException:



Public Member Functions

- [FareInputFileNotFoundException](#) (const std::string &iWhat)

22.9.1 Detailed Description

The fare input file can not be found.

Definition at line 111 of file [SIMFQT_Types.hpp](#).

22.9.2 Constructor & Destructor Documentation

22.9.2.1 SIMFQT::FareInputFileNotFoundException::FareInputFileNotFoundException (const std::string & *iWhat*)
[inline]

Constructor.

Definition at line 116 of file [SIMFQT_Types.hpp](#).

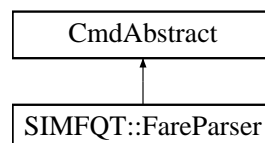
The documentation for this class was generated from the following file:

- [simfqt/SIMFQT_Types.hpp](#)

22.10 SIMFQT::FareParser Class Reference

```
#include <simfqt/command/FareParser.hpp>
```

Inheritance diagram for SIMFQT::FareParser:



Static Public Member Functions

- static void [fareRuleGeneration](#) (const [FareFilePath](#) &, stdair::BomRoot &)

22.10.1 Detailed Description

Class wrapping the parser entry point.

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

22.10.2 Member Function Documentation

22.10.2.1 void SIMFQT::FareParser::fareRuleGeneration (const FareFilePath & iFareFilename, stdair::BomRoot & ioBomRoot) [static]

Parses the CSV file describing the fares for the simulator, and generates the fare bom tree accordingly.

Parameters

<i>const</i>	FareFilePath & The file-name of the CSV-formatted fare input file.
<i>stdair::Bom-Root&</i>	Root of the BOM tree.

Definition at line 17 of file [FareParser.cpp](#).

References [SIMFQT::FareRuleFileParser::generateFareRules\(\)](#).

Referenced by [SIMFQT::SIMFQT_Service::parseAndLoad\(\)](#).

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

- [simfqt/command/FareParser.hpp](#)
- [simfqt/command/FareParser.cpp](#)

22.11 SIMFQT::FareQuoter Class Reference

Command wrapping the pricing request process.

```
#include <simfqt/command/FareQuoter.hpp>
```

Friends

- class [SIMFQT_Service](#)

22.11.1 Detailed Description

Command wrapping the pricing request process.

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

22.11.2 Friends And Related Function Documentation

22.11.2.1 friend class [SIMFQT_Service](#) [friend]

Friend classes: only the SimFQT service may access to the methods of that command class.

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

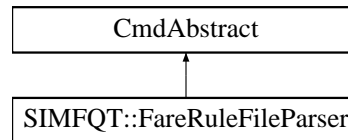
The documentation for this class was generated from the following files:

- [simfqt/command/FareQuoter.hpp](#)
- [simfqt/command/FareQuoter.cpp](#)

22.12 SIMFQT::FareRuleFileParser Class Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareRuleFileParser:



Public Member Functions

- [FareRuleFileParser](#) (stdair::BomRoot &ioBomRoot, const stdair::Filename_T &iFilename)
- void [generateFareRules](#) ()

22.12.1 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 254 of file [FareParserHelper.hpp](#).

22.12.2 Constructor & Destructor Documentation

22.12.2.1 SIMFQT::FareRuleFileParser::FareRuleFileParser (stdair::BomRoot & ioBomRoot, const stdair::Filename_T & iFilename)

Constructor.

Definition at line 642 of file [FareParserHelper.cpp](#).

22.12.3 Member Function Documentation

22.12.3.1 void SIMFQT::FareRuleFileParser::generateFareRules ()

Parse the input file and generate the fare rules.

Definition at line 664 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParser::fareRuleGeneration\(\)](#).

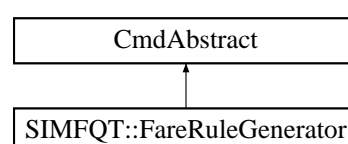
The documentation for this class was generated from the following files:

- simfqt/command/[FareParserHelper.hpp](#)
- simfqt/command/[FareParserHelper.cpp](#)

22.13 SIMFQT::FareRuleGenerator Class Reference

```
#include <simfqt/command/FareRuleGenerator.hpp>
```

Inheritance diagram for SIMFQT::FareRuleGenerator:



Friends

- class [FareFileParser](#)
- struct [FareParserHelper::doEndFare](#)
- class [FareParser](#)

22.13.1 Detailed Description

Class handling the generation / instantiation of the Fare BOM.

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

22.13.2 Friends And Related Function Documentation

22.13.2.1 friend class [FareFileParser](#) `[friend]`

Definition at line 38 of file [FareRuleGenerator.hpp](#).

22.13.2.2 friend struct [FareParserHelper::doEndFare](#) `[friend]`

Definition at line 39 of file [FareRuleGenerator.hpp](#).

22.13.2.3 friend class [FareParser](#) `[friend]`

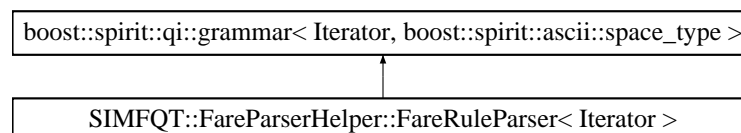
Definition at line 40 of file [FareRuleGenerator.hpp](#).

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

- [simfqt/command/FareRuleGenerator.hpp](#)
- [simfqt/command/FareRuleGenerator.cpp](#)

22.14 SIMFQT::FareParserHelper::FareRuleParser< Iterator > Struct Template Reference

Inheritance diagram for SIMFQT::FareParserHelper::FareRuleParser< Iterator >:



Public Member Functions

- [FareRuleParser](#) (stdair::BomRoot &ioBomRoot, [FareRuleStruct](#) &iofareRule)

Public Attributes

- [boost::spirit::qi::rule](#)
`< Iterator, boost::spirit::ascii::space_type >` [start](#)
- [boost::spirit::qi::rule](#)
`< Iterator, boost::spirit::ascii::space_type >` [comments](#)
- [boost::spirit::qi::rule](#)
`< Iterator, boost::spirit::ascii::space_type >` [fare_rule](#)

- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [fare_rule_end](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [fare_key](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [fare_id](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [origin](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [destination](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [tripType](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [dateRangeStart](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [dateRangeEnd](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [date](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [timeRangeStart](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [timeRangeEnd](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [time](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [point_of_sale](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [cabinCode](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [channel](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [advancePurchase](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [saturdayStay](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [changeFees](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [nonRefundable](#)

- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [minimumStay](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [fare](#)
- boost::spirit::qi::rule
< Iterator,
boost::spirit::ascii::space_type > [segment](#)
- stdair::BomRoot & [_bomRoot](#)
- [FareRuleStruct](#) & [_fareRule](#)

22.14.1 Detailed Description

template<typename Iterator>struct SIMFQT::FareParserHelper::FareRuleParser< Iterator >

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

fareID OriginCity (3-char airport code) DestinationCity (3-char airport code) DateRangeStart (yyyy-mm-dd) DateRangeEnd (yyyy-mm-dd) DepartureTimeRangeStart (hh:mm) DepartureTimeRangeEnd (hh:mm) POS (3-char point_of_sale city) Cabin Code (1-char cabin code) Channel (D=direct, I=indirect, N=online, F=offline) AdvancePurchase SaturdayNight (T=True, F=False) ChangeFees (T=True, F=False) NonRefundable (T=True, F=False) MinimumStay Price AirlineCode (2-char airline code) ClassList (List of 1-char class code) Grammar for the FareRule parser.

Definition at line 500 of file [FareParserHelper.cpp](#).

22.14.2 Constructor & Destructor Documentation

22.14.2.1 template<typename Iterator> SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser (stdair::BomRoot & *ioBomRoot*, FareRuleStruct & *iofareRule*) [inline]

Definition at line 504 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::_bomRoot](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::_fareRule](#), [SIMFQT::FareRuleStruct::_itDay](#), [SIMFQT::FareRuleStruct::_itHours](#), [SIMFQT::FareRuleStruct::_itMinutes](#), [SIMFQT::FareRuleStruct::_itMonth](#), [SIMFQT::FareRuleStruct::_itSeconds](#), [SIMFQT::FareRuleStruct::_itYear](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::advancePurchase](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::cabinCode](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::changeFees](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::channel](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::comments](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::date](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::dateRangeEnd](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::dateRangeStart](#), [SIMFQT::FareParserHelper::day_p](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::destination](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::fare](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::fare_id](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::fare_key](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::fare_rule](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::fare_rule_end](#), [SIMFQT::FareParserHelper::hour_p](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::minimumStay](#), [SIMFQT::FareParserHelper::minute_p](#), [SIMFQT::FareParserHelper::month_p](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::nonRefundable](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::origin](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::point_of_sale](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::saturdayStay](#), [SIMFQT::FareParserHelper::second_p](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::segment](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::start](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::time](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::timeRangeEnd](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::timeRangeStart](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::tripType](#), [SIMFQT::FareParserHelper::uint1_4_p](#), and [SIMFQT::FareParserHelper::year_p](#).

22.14.3 Member Data Documentation

22.14.3.1 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::start`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.2 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::comments`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.3 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::fare_rule`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.4 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::fare_rule_end`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.5 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::fare_key`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.6 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::fare_id`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.7 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::origin`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.8 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::destination`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.9 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::tripType`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.10 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::dateRangeStart`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.11 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::dateRangeEnd`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.12 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::date`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.13 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::timeRangeStart`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.14 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::timeRangeEnd`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.15 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::time`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.16 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::point_of_sale`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.17 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::cabinCode`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.18 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::channel`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.19 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::advancePurchase`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.20 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::saturdayStay`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.21 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::changeFees`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.22 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::nonRefundable`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.23 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::minimumStay`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.24 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::fare`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.25 `template<typename Iterator> boost::spirit::qi::rule<Iterator, boost::spirit::ascii::space_type>
SIMFQT::FareParserHelper::FareRuleParser< Iterator >::segment`

Definition at line 620 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.26 `template<typename Iterator> stdair::BomRoot& SIMFQT::FareParserHelper::FareRuleParser< Iterator
>::_bomRoot`

Definition at line 627 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.14.3.27 `template<typename Iterator> FareRuleStruct& SIMFQT::FareParserHelper::FareRuleParser< Iterator
>::_fareRule`

Definition at line 628 of file [FareParserHelper.cpp](#).

Referenced by [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

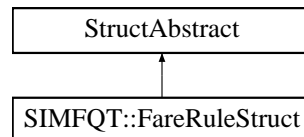
The documentation for this struct was generated from the following file:

- [simfqt/command/FareParserHelper.cpp](#)

22.15 SIMFQT::FareRuleStruct Struct Reference

```
#include <simfqt/bom/FareRuleStruct.hpp>
```

Inheritance diagram for SIMFQT::FareRuleStruct:



Public Member Functions

- [FareRuleStruct](#) ()
- [SIMFQT::FareQuoteID_T getFareID](#) () 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::DayDuration_T getAdvancePurchase](#) () const
- [stdair::SaturdayStay_T getSaturdayStay](#) () const
- [stdair::ChangeFees_T getChangeFees](#) () const
- [stdair::NonRefundable_T getNonRefundable](#) () const
- [stdair::DayDuration_T getMinimumStay](#) () const
- [stdair::PriceValue_T getFare](#) () 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 setFareID](#) (const [SIMFQT::FareQuoteID_T](#) &iFareQuoteID)
- [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 setAdvancePurchase](#) (const [stdair::DayDuration_T](#) &iAdvancePurchase)
- [void setSaturdayStay](#) (const [stdair::SaturdayStay_T](#) &iSaturdayStay)

- void [setChangeFees](#) (const stdair::ChangeFees_T &iChangeFees)
- void [setNonRefundable](#) (const stdair::NonRefundable_T &iNonRefundable)
- void [setMinimumStay](#) (const stdair::DayDuration_T &iMinimumStay)
- void [setFare](#) (const stdair::PriceValue_T &iFare)
- 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](#)

22.15.1 Detailed Description

Utility Structure for the parsing of fare-rule structures.

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

22.15.2 Constructor & Destructor Documentation

22.15.2.1 SIMFQT::FareRuleStruct::FareRuleStruct ()

Default constructor.

Definition at line 17 of file [FareRuleStruct.cpp](#).

22.15.3 Member Function Documentation

22.15.3.1 SIMFQT::FareQuoteID_T SIMFQT::FareRuleStruct::getFareID () const [inline]

Get the fare ID.

Definition at line 30 of file [FareRuleStruct.hpp](#).

22.15.3.2 stdair::AirportCode_T SIMFQT::FareRuleStruct::getOrigin () const [inline]

Get the origin.

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

Referenced by [SIMFQT::FareParserHelper::storePOS::operator\(\)](#).

22.15.3.3 stdair::AirportCode_T SIMFQT::FareRuleStruct::getDestination () const [inline]

Get the destination.

Definition at line 40 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storePOS::operator\(\)](#).

22.15.3.4 `stdair::TripType_T SIMFQT::FareRuleStruct::getTripType () const [inline]`

Get the trip type.

Definition at line 45 of file [FareRuleStruct.hpp](#).

22.15.3.5 `stdair::Date_T SIMFQT::FareRuleStruct::getDateRangeStart () const [inline]`

Get the date range start.

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

22.15.3.6 `stdair::Date_T SIMFQT::FareRuleStruct::getDateRangeEnd () const [inline]`

Get the date range end.

Definition at line 55 of file [FareRuleStruct.hpp](#).

22.15.3.7 `stdair::Duration_T SIMFQT::FareRuleStruct::getTimeRangeStart () const [inline]`

Get the time range start.

Definition at line 60 of file [FareRuleStruct.hpp](#).

22.15.3.8 `stdair::Duration_T SIMFQT::FareRuleStruct::getTimeRangeEnd () const [inline]`

Get the time range end.

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

22.15.3.9 `stdair::CabinCode_T SIMFQT::FareRuleStruct::getCabinCode () const [inline]`

Get the cabin code.

Definition at line 70 of file [FareRuleStruct.hpp](#).

22.15.3.10 `const stdair::CityCode_T SIMFQT::FareRuleStruct::getPOS () const [inline]`

Get the point-of-sale.

Definition at line 75 of file [FareRuleStruct.hpp](#).

22.15.3.11 `stdair::ChannelLabel_T SIMFQT::FareRuleStruct::getChannel () const [inline]`

Get the channel.

Definition at line 80 of file [FareRuleStruct.hpp](#).

22.15.3.12 `stdair::DayDuration_T SIMFQT::FareRuleStruct::getAdvancePurchase () const [inline]`

Get the advance purchase.

Definition at line 85 of file [FareRuleStruct.hpp](#).

22.15.3.13 `stdair::SaturdayStay_T SIMFQT::FareRuleStruct::getSaturdayStay () const [inline]`

Get the saturday stay option.

Definition at line 90 of file [FareRuleStruct.hpp](#).

22.15.3.14 `stdair::ChangeFees_T SIMFQT::FareRuleStruct::getChangeFees () const [inline]`

Get the change fees.

Definition at line 95 of file [FareRuleStruct.hpp](#).

22.15.3.15 `stdair::NonRefundable_T SIMFQT::FareRuleStruct::getNonRefundable () const [inline]`

Get the refundable option.

Definition at line 100 of file [FareRuleStruct.hpp](#).

22.15.3.16 `stdair::DayDuration_T SIMFQT::FareRuleStruct::getMinimumStay () const [inline]`

Get the minimum stay.

Definition at line 105 of file [FareRuleStruct.hpp](#).

22.15.3.17 `stdair::PriceValue_T SIMFQT::FareRuleStruct::getFare () const [inline]`

Get the fare.

Definition at line 110 of file [FareRuleStruct.hpp](#).

22.15.3.18 `stdair::AirlineCode_T SIMFQT::FareRuleStruct::getAirlineCode () const [inline]`

Get the airline code.

Definition at line 115 of file [FareRuleStruct.hpp](#).

22.15.3.19 `stdair::ClassCode_T SIMFQT::FareRuleStruct::getClassCode () const [inline]`

Get the class code.

Definition at line 120 of file [FareRuleStruct.hpp](#).

22.15.3.20 `const unsigned int SIMFQT::FareRuleStruct::getAirlineListSize () const [inline]`

Get the size of the airline code list.

Definition at line 125 of file [FareRuleStruct.hpp](#).

22.15.3.21 `const unsigned int SIMFQT::FareRuleStruct::getClassCodeListSize () const [inline]`

Get the size of the class code list.

Definition at line 130 of file [FareRuleStruct.hpp](#).

22.15.3.22 `stdair::AirlineCodeList_T SIMFQT::FareRuleStruct::getAirlineList () const [inline]`

Get the airline code list.

Definition at line 135 of file [FareRuleStruct.hpp](#).

22.15.3.23 `stdair::ClassList_StringList_T SIMFQT::FareRuleStruct::getClassCodeList () const [inline]`

Get the class code list.

Definition at line 140 of file [FareRuleStruct.hpp](#).

22.15.3.24 `stdair::Date_T SIMFQT::FareRuleStruct::calculateDate () const`

Calculate the date from the staging details.

Definition at line 39 of file [FareRuleStruct.cpp](#).

References [_itDay](#), [_itMonth](#), and [_itYear](#).

Referenced by [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)\(\)](#), and [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)\(\)](#).

22.15.3.25 `stdair::Duration_T SIMFQT::FareRuleStruct::calculateTime () const`

Calculate the time from the staging details.

Definition at line 45 of file [FareRuleStruct.cpp](#).

References [_itHours](#), [_itMinutes](#), and [_itSeconds](#).

Referenced by [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)\(\)](#), and [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)\(\)](#).

22.15.3.26 `const std::string SIMFQT::FareRuleStruct::describe () const`

Display of the structure.

Definition at line 54 of file [FareRuleStruct.cpp](#).

Referenced by [SIMFQT::FareParserHelper::doEndFare::operator\(\)\(\)](#).

22.15.3.27 `void SIMFQT::FareRuleStruct::setFareID (const SIMFQT::FareQuoteID_T & iFareQuoteID) [inline]`

Set the fare ID.

Definition at line 158 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)\(\)](#).

22.15.3.28 `void SIMFQT::FareRuleStruct::setOrigin (const stdair::AirportCode_T & iOrigin) [inline]`

Set the origin.

Definition at line 163 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeOrigin::operator\(\)\(\)](#).

22.15.3.29 `void SIMFQT::FareRuleStruct::setDestination (const stdair::AirportCode_T & iDestination) [inline]`

Set the destination.

Definition at line 168 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeDestination::operator\(\)\(\)](#).

22.15.3.30 `void SIMFQT::FareRuleStruct::setTripType (const stdair::TripType_T & iTripType) [inline]`

Set the trip type.

Definition at line 173 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeTripType::operator\(\)\(\)](#).

22.15.3.31 `void SIMFQT::FareRuleStruct::setDateRangeStart (const stdair::Date_T & iDateRangeStart) [inline]`

Set the date range start.

Definition at line 178 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)\(\)](#).

22.15.3.32 `void SIMFQT::FareRuleStruct::setDateRangeEnd (const stdair::Date_T & iDateRangeEnd) [inline]`

Set the date range end.

Definition at line 183 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)\(\)](#).

22.15.3.33 void SIMFQT::FareRuleStruct::setTimeRangeStart (const stdair::Duration_T & *iTimeRangeStart*) [inline]

Set the time range start.

Definition at line 188 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)\(\)](#).

22.15.3.34 void SIMFQT::FareRuleStruct::setTimeRangeEnd (const stdair::Duration_T & *iTimeRangeEnd*) [inline]

Set the time range end.

Definition at line 193 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)\(\)](#).

22.15.3.35 void SIMFQT::FareRuleStruct::setCabinCode (const stdair::CabinCode_T & *iCabinCode*) [inline]

Set the cabin code.

Definition at line 198 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)\(\)](#).

22.15.3.36 void SIMFQT::FareRuleStruct::setPOS (const stdair::CityCode_T & *iPOS*) [inline]

Set the point-of-sale.

Definition at line 203 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storePOS::operator\(\)\(\)](#).

22.15.3.37 void SIMFQT::FareRuleStruct::setChannel (const stdair::ChannelLabel_T & *iChannel*) [inline]

Set the channel.

Definition at line 208 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeChannel::operator\(\)\(\)](#).

22.15.3.38 void SIMFQT::FareRuleStruct::setAdvancePurchase (const stdair::DayDuration_T & *iAdvancePurchase*)
[inline]

Set the advance purchase.

Definition at line 213 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)\(\)](#).

22.15.3.39 void SIMFQT::FareRuleStruct::setSaturdayStay (const stdair::SaturdayStay_T & *iSaturdayStay*) [inline]

Set the saturday stay option.

Definition at line 218 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)\(\)](#).

22.15.3.40 void SIMFQT::FareRuleStruct::setChangeFees (const stdair::ChangeFees_T & *iChangeFees*) [inline]

Set the change fees.

Definition at line 223 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)\(\)](#).

22.15.3.41 void SIMFQT::FareRuleStruct::setNonRefundable (const stdair::NonRefundable_T & *iNonRefundable*)
[inline]

Set the refundable option.

Definition at line 228 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#).

22.15.3.42 void SIMFQT::FareRuleStruct::setMinimumStay (const stdair::DayDuration_T & *iMinimumStay*) [inline]

Set the minimum stay.

Definition at line 233 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#).

22.15.3.43 void SIMFQT::FareRuleStruct::setFare (const stdair::PriceValue_T & *iFare*) [inline]

Set the fare.

Definition at line 238 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeFare::operator\(\)](#).

22.15.3.44 void SIMFQT::FareRuleStruct::setAirlineCode (const stdair::AirlineCode_T & *iAirlineCode*) [inline]

Set the airline code.

Definition at line 243 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#).

22.15.3.45 void SIMFQT::FareRuleStruct::setClassCode (const stdair::ClassCode_T & *iClassCode*) [inline]

Set the class code.

Definition at line 248 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#).

22.15.3.46 void SIMFQT::FareRuleStruct::clearAirlineCodeList () [inline]

Empty the airline code list.

Definition at line 253 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#).

22.15.3.47 void SIMFQT::FareRuleStruct::clearClassCodeList () [inline]

Empty the class code list.

Definition at line 258 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#).

22.15.3.48 void SIMFQT::FareRuleStruct::addAirlineCode (const stdair::AirlineCode_T & *iAirlineCode*) [inline]

Add an airline code to the list.

Definition at line 263 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#).

22.15.3.49 void SIMFQT::FareRuleStruct::addClassCode (const stdair::ClassCode_T & *iClassCode*) [inline]

Add a class code to the list.

Definition at line 268 of file [FareRuleStruct.hpp](#).

Referenced by [SIMFQT::FareParserHelper::storeClass::operator\(\)](#).

22.15.4 Member Data Documentation

22.15.4.1 stdair::year_t SIMFQT::FareRuleStruct::_itYear

Staging Date.

Definition at line 275 of file [FareRuleStruct.hpp](#).

Referenced by [calculateDate\(\)](#), and [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.15.4.2 stdair::month_t SIMFQT::FareRuleStruct::_itMonth

Definition at line 276 of file [FareRuleStruct.hpp](#).

Referenced by [calculateDate\(\)](#), and [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.15.4.3 stdair::day_t SIMFQT::FareRuleStruct::_itDay

Definition at line 277 of file [FareRuleStruct.hpp](#).

Referenced by [calculateDate\(\)](#), and [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.15.4.4 stdair::hour_t SIMFQT::FareRuleStruct::_itHours

Staging Time.

Definition at line 280 of file [FareRuleStruct.hpp](#).

Referenced by [calculateTime\(\)](#), and [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.15.4.5 stdair::minute_t SIMFQT::FareRuleStruct::_itMinutes

Definition at line 281 of file [FareRuleStruct.hpp](#).

Referenced by [calculateTime\(\)](#), and [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#).

22.15.4.6 stdair::second_t SIMFQT::FareRuleStruct::_itSeconds

Definition at line 282 of file [FareRuleStruct.hpp](#).

Referenced by [calculateTime\(\)](#), [SIMFQT::FareParserHelper::FareRuleParser< Iterator >::FareRuleParser\(\)](#), [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), and [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#).

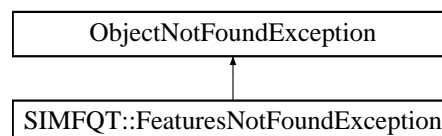
The documentation for this struct was generated from the following files:

- [simfqt/bom/FareRuleStruct.hpp](#)
- [simfqt/bom/FareRuleStruct.cpp](#)

22.16 SIMFQT::FeaturesNotFoundException Class Reference

```
#include <simfqt/SIMFQT_Types.hpp>
```

Inheritance diagram for SIMFQT::FeaturesNotFoundException:



Public Member Functions

- [FeaturesNotFoundException](#) (const std::string &iWhat)

22.16.1 Detailed Description

The fare features can not be found.

Definition at line 87 of file [SIMFQT_Types.hpp](#).

22.16.2 Constructor & Destructor Documentation

22.16.2.1 `SIMFQT::FeaturesNotFoundException::FeaturesNotFoundException (const std::string & iWhat) [inline]`

Constructor.

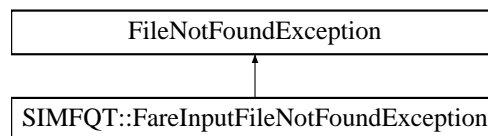
Definition at line 92 of file [SIMFQT_Types.hpp](#).

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

- [simfqt/SIMFQT_Types.hpp](#)

22.17 FileNotFoundException Class Reference

Inheritance diagram for FileNotFoundException:



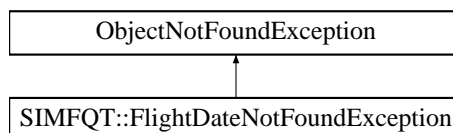
The documentation for this class was generated from the following file:

- [simfqt/SIMFQT_Types.hpp](#)

22.18 SIMFQT::FlightDateNotFoundException Class Reference

```
#include <simfqt/SIMFQT_Types.hpp>
```

Inheritance diagram for SIMFQT::FlightDateNotFoundException:



Public Member Functions

- [FlightDateNotFoundException](#) (const std::string &iWhat)

22.18.1 Detailed Description

The departure date of the flight can not be found.

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

22.18.2 Constructor & Destructor Documentation

22.18.2.1 SIMFQT::FlightDateNotFoundException::FlightDateNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 68 of file [SIMFQT_Types.hpp](#).

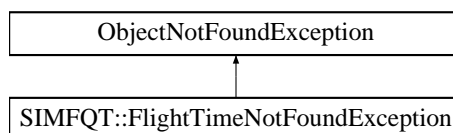
The documentation for this class was generated from the following file:

- [simfqt/SIMFQT_Types.hpp](#)

22.19 SIMFQT::FlightTimeNotFoundException Class Reference

```
#include <simfqt/SIMFQT_Types.hpp>
```

Inheritance diagram for SIMFQT::FlightTimeNotFoundException:



Public Member Functions

- [FlightTimeNotFoundException](#) (const std::string &iWhat)

22.19.1 Detailed Description

The departure time of the flight can not be found.

Definition at line 75 of file [SIMFQT_Types.hpp](#).

22.19.2 Constructor & Destructor Documentation

22.19.2.1 SIMFQT::FlightTimeNotFoundException::FlightTimeNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

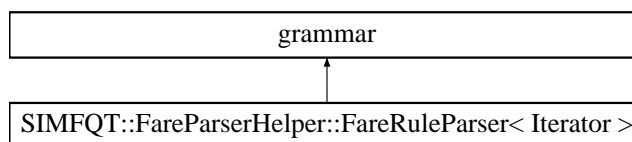
Definition at line 80 of file [SIMFQT_Types.hpp](#).

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

- [simfqt/SIMFQT_Types.hpp](#)

22.20 grammar Class Reference

Inheritance diagram for grammar:

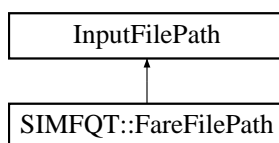


The documentation for this class was generated from the following file:

- [simfqt/command/FareParserHelper.cpp](#)

22.21 InputFilePath Class Reference

Inheritance diagram for InputFilePath:



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

- [simfqt/SIMFQT_Types.hpp](#)

22.22 ObjectNotFoundException Class Reference

Inheritance diagram for ObjectNotFoundException:



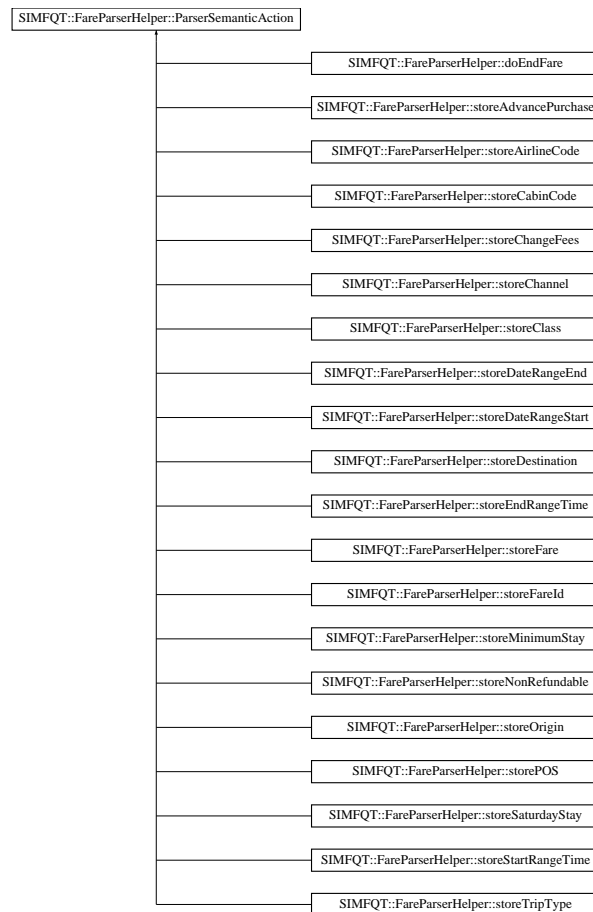
The documentation for this class was generated from the following file:

- [simfqt/SIMFQT_Types.hpp](#)

22.23 SIMFQT::FareParserHelper::ParserSemanticAction Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::ParserSemanticAction:



Public Member Functions

- [ParserSemanticAction](#) ([FareRuleStruct](#) &)

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.23.1 Detailed Description

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

Definition at line 31 of file [FareParserHelper.hpp](#).

22.23.2 Constructor & Destructor Documentation

22.23.2.1 SIMFQT::FareParserHelper::ParserSemanticAction::ParserSemanticAction ([FareRuleStruct](#) & *ioFareRule*)

Actor Constructor.

Definition at line 29 of file [FareParserHelper.cpp](#).

22.23.3 Member Data Documentation

22.23.3.1 FareRuleStruct& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule

Actor Context.

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

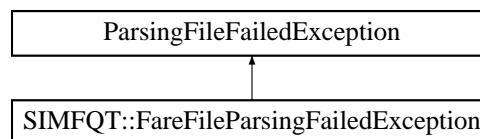
Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

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

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.24 ParsingFileFailedException Class Reference

Inheritance diagram for ParsingFileFailedException:



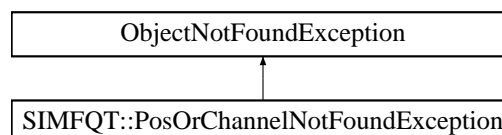
The documentation for this class was generated from the following file:

- [simfqt/SIMFQT_Types.hpp](#)

22.25 SIMFQT::PosOrChannelNotFoundException Class Reference

```
#include <simfqt/SIMFQT_Types.hpp>
```

Inheritance diagram for SIMFQT::PosOrChannelNotFoundException:



Public Member Functions

- [PosOrChannelNotFoundException](#) (const std::string &iWhat)

22.25.1 Detailed Description

The given POS/channel can not be found.

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

22.25.2 Constructor & Destructor Documentation

22.25.2.1 SIMFQT::PosOrChannelNotFoundException::PosOrChannelNotFoundException (const std::string & *iWhat*)
[inline]

Constructor.

Definition at line 56 of file [SIMFQT_Types.hpp](#).

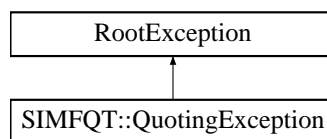
The documentation for this class was generated from the following file:

- [simfqt/SIMFQT_Types.hpp](#)

22.26 SIMFQT::QuotingException Class Reference

```
#include <simfqt/SIMFQT_Types.hpp>
```

Inheritance diagram for SIMFQT::QuotingException:



22.26.1 Detailed Description

The pricing operation fails.

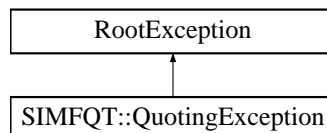
Definition at line 123 of file [SIMFQT_Types.hpp](#).

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

- [simfqt/SIMFQT_Types.hpp](#)

22.27 RootException Class Reference

Inheritance diagram for RootException:

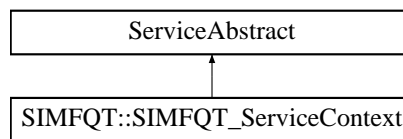


The documentation for this class was generated from the following file:

- [simfqt/SIMFQT_Types.hpp](#)

22.28 ServiceAbstract Class Reference

Inheritance diagram for ServiceAbstract:



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

- [simfqt/service/SIMFQT_ServiceContext.hpp](#)

22.29 SIMFQT::SIMFQT_Service Class Reference

Interface for the [SIMFQT](#) Services.

```
#include <simfqt/SIMFQT_Service.hpp>
```

Public Member Functions

- [SIMFQT_Service](#) (const stdair::BasLogParams &)
- [SIMFQT_Service](#) (const stdair::BasLogParams &, const stdair::BasDBParams &)
- [SIMFQT_Service](#) (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr)
- void [parseAndLoad](#) (const [FareFilePath](#) &iFareFilename)
- [~SIMFQT_Service](#) ()
- void [buildSampleBom](#) ()
- stdair::BookingRequestStruct [buildBookingRequest](#) (const bool isForCRS=false)
- void [buildSampleTravelSolutions](#) (stdair::TravelSolutionList_T &)
- void [quotePrices](#) (const stdair::BookingRequestStruct &, stdair::TravelSolutionList_T &)
- std::string [csvDisplay](#) () const
- std::string [csvDisplay](#) (const stdair::TravelSolutionList_T &) const
- std::string [csvDisplay](#) (const stdair::AirportCode_T &ioOrigin, const stdair::AirportCode_T &ioDestination, const stdair::Date_T &ioDepartureDate) const
- std::string [list](#) () const
- bool [check](#) (const stdair::AirportCode_T &ioOrigin, const stdair::AirportCode_T &ioDestination, const stdair::Date_T &ioDepartureDate) const

22.29.1 Detailed Description

Interface for the [SIMFQT](#) Services.

Definition at line 31 of file [SIMFQT_Service.hpp](#).

22.29.2 Constructor & Destructor Documentation

22.29.2.1 SIMFQT::SIMFQT_Service::SIMFQT_Service (const stdair::BasLogParams & iLogParams)

Constructor.

The `initSimfqtService()` 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

<i>const</i>	stdair::BasLogParams& Parameters for the output log stream.
--------------	-------------------------------------------------------------

Definition at line 36 of file [SIMFQT_Service.cpp](#).

22.29.2.2 SIMFQT::SIMFQT_Service::SIMFQT_Service (const stdair::BasLogParams & *iLogParams*, const stdair::BasDBParams & *iDBParams*)

Constructor.

The `initSimfqtService()` 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

<i>const</i>	stdair::BasLogParams& Parameters for the output log stream.
<i>const</i>	stdair::BasDBParams& Parameters for the database access.

Definition at line 56 of file [SIMFQT_Service.cpp](#).

22.29.2.3 SIMFQT::SIMFQT_Service::SIMFQT_Service (stdair::STDAIR_ServicePtr_T *ioSTDAIR_ServicePtr*)

Constructor.

The `initSimfqtService()` 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 [SIMFQT_Service](#) is itself being initialised by another library service such as [SIMCRS_Service](#)).

Parameters

<i>stdair::STDAIR_ServicePtr_T</i>	Reference on the STDAIR service.
------------------------------------	----------------------------------

Definition at line 78 of file [SIMFQT_Service.cpp](#).

22.29.2.4 SIMFQT::SIMFQT_Service::~~SIMFQT_Service ()

Destructor.

Definition at line 94 of file [SIMFQT_Service.cpp](#).

22.29.3 Member Function Documentation

22.29.3.1 void SIMFQT::SIMFQT_Service::parseAndLoad (const FareFilePath & *iFareFilename*)

Parse the fare dump and load it into memory.

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

Parameters

<i>const</i>	FareFilePath & Filename of the input fare file.
--------------	-----------------------------------------------------------------

Definition at line 171 of file [SIMFQT_Service.cpp](#).

References [SIMFQT::FareParser::fareRuleGeneration\(\)](#).

Referenced by [main\(\)](#).

22.29.3.2 void SIMFQT::SIMFQT_Service::buildSampleBom ()

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

As for now, two sample BOM trees can be built.

- One BOM tree is based on two actual inventories (one for BA, another for AF). Each inventory contains one flight. One of those flights has two legs (and therefore three segments).
- The other BOM tree is fake, as a hook for RMOL to work.

Definition at line 185 of file [SIMFQT_Service.cpp](#).

Referenced by [main\(\)](#).

22.29.3.3 stdair::BookingRequestStruct SIMFQT::SIMFQT_Service::buildBookingRequest (const bool *isForCRS* = false)

Build a BookingRequest structure (for test purposes).

Returns

stdair::BookingRequestStruct The created BookingRequest structure.

Definition at line 231 of file [SIMFQT_Service.cpp](#).

Referenced by [main\(\)](#).

22.29.3.4 void SIMFQT::SIMFQT_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

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

Definition at line 255 of file [SIMFQT_Service.cpp](#).

Referenced by [main\(\)](#).

22.29.3.5 void SIMFQT::SIMFQT_Service::quotePrices (const stdair::BookingRequestStruct & *iBookingRequest*, stdair::TravelSolutionList_T & *ioTravelSolutionList*)

Calculate the prices for a given list of travel solutions.

A stdair::Fare_T attribute is calculated for every travel solution of the list.

Parameters

<i>stdair::Booking-RequestStruct&</i>	Booking request.
<i>stdair::Travel-SolutionList_T&</i>	List of travel solution.

Definition at line 391 of file [SIMFQT_Service.cpp](#).

Referenced by [main\(\)](#).

22.29.3.6 std::string SIMFQT::SIMFQT_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 276 of file [SIMFQT_Service.cpp](#).

Referenced by [main\(\)](#).

22.29.3.7 std::string SIMFQT::SIMFQT_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 303 of file [SIMFQT_Service.cpp](#).

22.29.3.8 std::string SIMFQT::SIMFQT_Service::csvDisplay (const stdair::AirportCode_T & ioOrigin, const stdair::AirportCode_T & ioDestination, const stdair::Date_T & ioDepartureDate) const

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

Parameters

<i>const</i>	stdair::AirportCode_T& Origin airport of the fare-rules to display
<i>const</i>	stdair::AirportCode_T& Destination airport of the fare- rules to display.
<i>const</i>	stdair::Date_T& Departure date of the fare-rules to display.

Returns

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

Definition at line 325 of file [SIMFQT_Service.cpp](#).

22.29.3.9 std::string SIMFQT::SIMFQT_Service::list () const

Display (dump in the returned string) the airport pairs and the corresponding departure dates of the fare rules stored in the BOM tree.

Returns

std::string Output string in which the airport pairs and departure dates are logged/dumped.

Definition at line 348 of file [SIMFQT_Service.cpp](#).

22.29.3.10 `bool SIMFQT::SIMFQT_Service::check (const stdair::AirportCode_T & ioOrigin, const stdair::AirportCode_T & ioDestination, const stdair::Date_T & ioDepartureDate) const`

Check whether the given couple airportpair-date is a valid one.

Parameters

<i>const</i>	stdair::AirportCode_T& Origin airport of the fare rule to check.
<i>const</i>	stdair::AirportCode_T& Destination airport of the fare rule to check.
<i>const</i>	stdair::Date_T& Departure date of the fare rule to check.

Returns

bool Whether or not the given airportpair-date couple is a valid one.

Definition at line 369 of file [SIMFQT_Service.cpp](#).

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

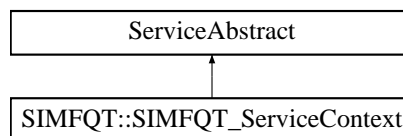
- [simfqt/SIMFQT_Service.hpp](#)
- [simfqt/service/SIMFQT_Service.cpp](#)

22.30 SIMFQT::SIMFQT_ServiceContext Class Reference

Class holding the context of the SimFQT services.

```
#include <simfqt/service/SIMFQT_ServiceContext.hpp>
```

Inheritance diagram for SIMFQT::SIMFQT_ServiceContext:



Friends

- class [SIMFQT_Service](#)
- class [FacSimfqtServiceContext](#)

22.30.1 Detailed Description

Class holding the context of the SimFQT services.

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

22.30.2 Friends And Related Function Documentation

22.30.2.1 friend class SIMFQT_Service [friend]

The [SIMFQT_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 [SIMFQT_ServiceContext.hpp](#).

22.30.2.2 friend class FacSimfqtServiceContext [friend]

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

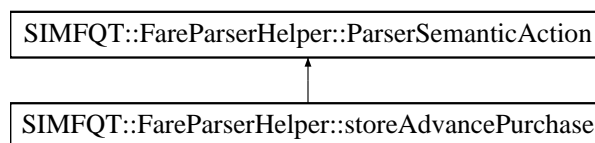
The documentation for this class was generated from the following files:

- [simfqt/service/SIMFQT_ServiceContext.hpp](#)
- [simfqt/service/SIMFQT_ServiceContext.cpp](#)

22.31 SIMFQT::FareParserHelper::storeAdvancePurchase Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeAdvancePurchase:



Public Member Functions

- [storeAdvancePurchase](#) ([FareRuleStruct](#) &)
- void [operator\(\)](#) (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.31.1 Detailed Description

Store the parsed advance purchase days.

Definition at line 150 of file [FareParserHelper.hpp](#).

22.31.2 Constructor & Destructor Documentation

22.31.2.1 SIMFQT::FareParserHelper::storeAdvancePurchase::storeAdvancePurchase ([FareRuleStruct](#) & *ioFareRule*)

Actor Constructor.

Definition at line 251 of file [FareParserHelper.cpp](#).

22.31.3 Member Function Documentation

22.31.3.1 void SIMFQT::FareParserHelper::storeAdvancePurchase::operator() (unsigned int *iAdvancePurchase*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 256 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::setAdvancePurchase\(\)](#).

22.31.4 Member Data Documentation

22.31.4.1 FareRuleStruct& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

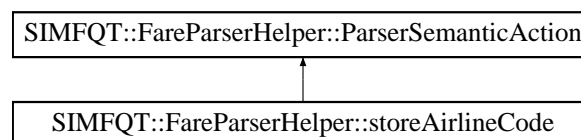
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.32 SIMFQT::FareParserHelper::storeAirlineCode Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeAirlineCode:



Public Member Functions

- [storeAirlineCode](#) ([FareRuleStruct](#) &)
- void [operator\(\)](#) (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.32.1 Detailed Description

Store the parsed airline code.

Definition at line 210 of file [FareParserHelper.hpp](#).

22.32.2 Constructor & Destructor Documentation

22.32.2.1 SIMFQT::FareParserHelper::storeAirlineCode::storeAirlineCode ([FareRuleStruct](#) & *ioFareRule*)

Actor Constructor.

Definition at line 375 of file [FareParserHelper.cpp](#).

22.32.3 Member Function Documentation

22.32.3.1 void SIMFQT::FareParserHelper::storeAirlineCode::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 380 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::add-AirlineCode\(\)](#).

22.32.4 Member Data Documentation

22.32.4.1 FareRuleStruct& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

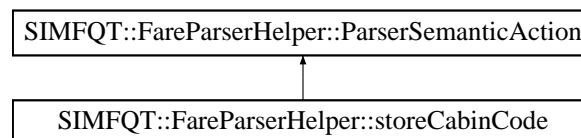
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.33 SIMFQT::FareParserHelper::storeCabinCode Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeCabinCode:



Public Member Functions

- [storeCabinCode](#) ([FareRuleStruct](#) &)
- void [operator\(\)](#) (char, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.33.1 Detailed Description

Store the cabin code.

Definition at line 130 of file [FareParserHelper.hpp](#).

22.33.2 Constructor & Destructor Documentation

22.33.2.1 SIMFQT::FareParserHelper::storeCabinCode::storeCabinCode ([FareRuleStruct](#) & [ioFareRule](#))

Actor Constructor.

Definition at line 209 of file [FareParserHelper.cpp](#).

22.33.3 Member Function Documentation

22.33.3.1 void SIMFQT::FareParserHelper::storeCabinCode::operator() ([char iChar](#), [boost::spirit::qi::unused_type](#) , [boost::spirit::qi::unused_type](#)) const

Actor Function (functor).

Definition at line 214 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::setCabinCode\(\)](#).

22.33.4 Member Data Documentation

22.33.4.1 [FareRuleStruct](#)& [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#) [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

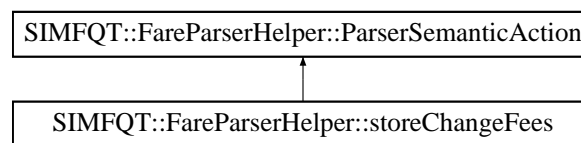
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.34 SIMFQT::FareParserHelper::storeChangeFees Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeChangeFees:



Public Member Functions

- [storeChangeFees](#) ([FareRuleStruct](#) &)
- void [operator\(\)](#) (char, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.34.1 Detailed Description

Store the parsed change fees.

Definition at line 170 of file [FareParserHelper.hpp](#).

22.34.2 Constructor & Destructor Documentation

22.34.2.1 SIMFQT::FareParserHelper::storeChangeFees::storeChangeFees ([FareRuleStruct](#) & *ioFareRule*)

Actor Constructor.

Definition at line 292 of file [FareParserHelper.cpp](#).

22.34.3 Member Function Documentation

22.34.3.1 void SIMFQT::FareParserHelper::storeChangeFees::operator() (char *iChangefees*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 297 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::setChangeFees\(\)](#).

22.34.4 Member Data Documentation

22.34.4.1 [FareRuleStruct](#)& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#),

[SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

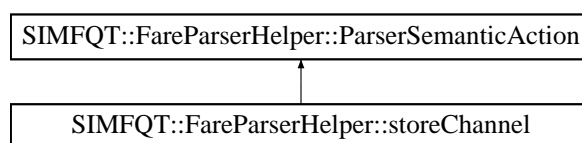
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.35 SIMFQT::FareParserHelper::storeChannel Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeChannel:



Public Member Functions

- [storeChannel](#) ([FareRuleStruct](#) &)
- void [operator\(\)](#) (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.35.1 Detailed Description

Store the channel distribution.

Definition at line 140 of file [FareParserHelper.hpp](#).

22.35.2 Constructor & Destructor Documentation

22.35.2.1 SIMFQT::FareParserHelper::storeChannel::storeChannel (FareRuleStruct & ioFareRule)

Actor Constructor.

Definition at line 230 of file [FareParserHelper.cpp](#).

22.35.3 Member Function Documentation

22.35.3.1 void SIMFQT::FareParserHelper::storeChannel::operator() (std::vector< char > iChar, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 235 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::setChannel\(\)](#).

22.35.4 Member Data Documentation

22.35.4.1 FareRuleStruct& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

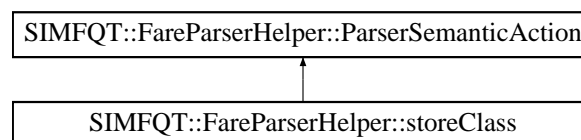
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.36 SIMFQT::FareParserHelper::storeClass Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeClass:



Public Member Functions

- [storeClass](#) ([FareRuleStruct](#) &)
- void [operator\(\)](#) (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.36.1 Detailed Description

Store the parsed class code.

Definition at line 220 of file [FareParserHelper.hpp](#).

22.36.2 Constructor & Destructor Documentation

22.36.2.1 SIMFQT::FareParserHelper::storeClass::storeClass ([FareRuleStruct](#) & *ioFareRule*)

Actor Constructor.

Definition at line 393 of file [FareParserHelper.cpp](#).

22.36.3 Member Function Documentation

22.36.3.1 void SIMFQT::FareParserHelper::storeClass::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 398 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::add-ClassCode\(\)](#).

22.36.4 Member Data Documentation

22.36.4.1 FareRuleStruct& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

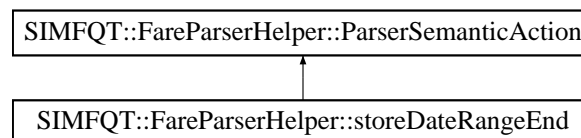
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.37 SIMFQT::FareParserHelper::storeDateRangeEnd Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeDateRangeEnd:



Public Member Functions

- [storeDateRangeEnd](#) (FareRuleStruct &)
- void [operator\(\)](#) (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.37.1 Detailed Description

Store the parsed end of the date range.

Definition at line 90 of file [FareParserHelper.hpp](#).

22.37.2 Constructor & Destructor Documentation

22.37.2.1 SIMFQT::FareParserHelper::storeDateRangeEnd::storeDateRangeEnd ([FareRuleStruct](#) & [ioFareRule](#))

Actor Constructor.

Definition at line 128 of file [FareParserHelper.cpp](#).

22.37.3 Member Function Documentation

22.37.3.1 void SIMFQT::FareParserHelper::storeDateRangeEnd::operator() ([boost::spirit::qi::unused_type](#) , [boost::spirit::qi::unused_type](#) , [boost::spirit::qi::unused_type](#)) const

Actor Function (functor).

Definition at line 133 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), [SIMFQT::FareRuleStruct::calculateDate\(\)](#), and [SIMFQT::FareRuleStruct::setDateRangeEnd\(\)](#).

22.37.4 Member Data Documentation

22.37.4.1 [FareRuleStruct](#)& [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#) [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

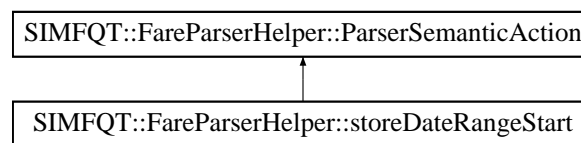
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.38 SIMFQT::FareParserHelper::storeDateRangeStart Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeDateRangeStart:



Public Member Functions

- [storeDateRangeStart](#) ([FareRuleStruct](#) &)
- void [operator\(\)](#) ([boost::spirit::qi::unused_type](#), [boost::spirit::qi::unused_type](#), [boost::spirit::qi::unused_type](#)) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.38.1 Detailed Description

Store the parsed start of the date range.

Definition at line 80 of file [FareParserHelper.hpp](#).

22.38.2 Constructor & Destructor Documentation

22.38.2.1 SIMFQT::FareParserHelper::storeDateRangeStart::storeDateRangeStart ([FareRuleStruct](#) & *ioFareRule*)

Actor Constructor.

Definition at line 112 of file [FareParserHelper.cpp](#).

22.38.3 Member Function Documentation

22.38.3.1 void SIMFQT::FareParserHelper::storeDateRangeStart::operator() ([boost::spirit::qi::unused_type](#) , [boost::spirit::qi::unused_type](#) , [boost::spirit::qi::unused_type](#)) const

Actor Function (functor).

Definition at line 117 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), [SIMFQT::FareRuleStruct::calculateDate\(\)](#), and [SIMFQT::FareRuleStruct::setDateRangeStart\(\)](#).

22.38.4 Member Data Documentation

22.38.4.1 [FareRuleStruct](#)& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), and [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#).

[SIMFQT::FareParserHelper::storeChannel::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#)(), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

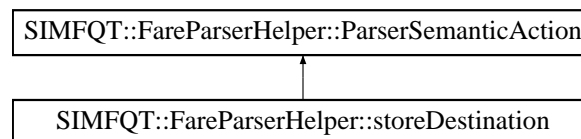
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.39 SIMFQT::FareParserHelper::storeDestination Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeDestination:



Public Member Functions

- [storeDestination](#) ([FareRuleStruct](#) &)
- [operator](#)() (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.39.1 Detailed Description

Store the parsed destination.

Definition at line 59 of file [FareParserHelper.hpp](#).

22.39.2 Constructor & Destructor Documentation

22.39.2.1 SIMFQT::FareParserHelper::storeDestination::storeDestination ([FareRuleStruct](#) & *ioFareRule*)

Actor Constructor.

Definition at line 74 of file [FareParserHelper.cpp](#).

22.39.3 Member Function Documentation

22.39.3.1 void SIMFQT::FareParserHelper::storeDestination::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 79 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::set-Destination\(\)](#).

22.39.4 Member Data Documentation

22.39.4.1 FareRuleStruct& SIMFQT::FareParserHelper::ParserSemanticAction::fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

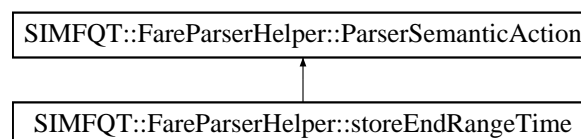
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.40 SIMFQT::FareParserHelper::storeEndRangeTime Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeEndRangeTime:



Public Member Functions

- [storeEndRangeTime](#) ([FareRuleStruct](#) &)
- [void operator\(\)](#) ([boost::spirit::qi::unused_type](#), [boost::spirit::qi::unused_type](#), [boost::spirit::qi::unused_type](#)) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.40.1 Detailed Description

Store the parsed end range time.

Definition at line 110 of file [FareParserHelper.hpp](#).

22.40.2 Constructor & Destructor Documentation

22.40.2.1 SIMFQT::FareParserHelper::storeEndRangeTime::storeEndRangeTime (FareRuleStruct & ioFareRule)

Actor Constructor.

Definition at line 167 of file [FareParserHelper.cpp](#).

22.40.3 Member Function Documentation

22.40.3.1 void SIMFQT::FareParserHelper::storeEndRangeTime::operator() (boost::spirit::qi::unused_type , boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 172 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), [SIMFQT::FareRuleStruct::_itSeconds](#), [SIMFQT::FareRuleStruct::calculateTime\(\)](#), and [SIMFQT::FareRuleStruct::setTimeRangeEnd\(\)](#).

22.40.4 Member Data Documentation

22.40.4.1 FareRuleStruct& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

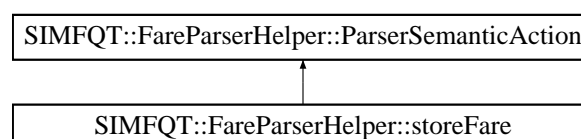
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.41 SIMFQT::FareParserHelper::storeFare Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeFare:



Public Member Functions

- [storeFare](#) ([FareRuleStruct](#) &)
- void [operator](#)() (double, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.41.1 Detailed Description

Store the parsed fare value.

Definition at line 200 of file [FareParserHelper.hpp](#).

22.41.2 Constructor & Destructor Documentation

22.41.2.1 SIMFQT::FareParserHelper::storeFare::storeFare ([FareRuleStruct](#) & *ioFareRule*)

Actor Constructor.

Definition at line 359 of file [FareParserHelper.cpp](#).

22.41.3 Member Function Documentation

22.41.3.1 void SIMFQT::FareParserHelper::storeFare::operator() (double *iFare*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 364 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::setFare\(\)](#).

22.41.4 Member Data Documentation

22.41.4.1 [FareRuleStruct](#)& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

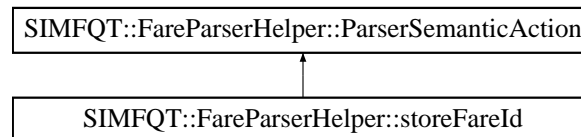
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.42 SIMFQT::FareParserHelper::storeFareId Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeFareId:



Public Member Functions

- [storeFareId](#) ([FareRuleStruct](#) &)
- void [operator](#)() (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.42.1 Detailed Description

Store the parsed fare Id.

Definition at line 39 of file [FareParserHelper.hpp](#).

22.42.2 Constructor & Destructor Documentation

22.42.2.1 SIMFQT::FareParserHelper::storeFareId::storeFareId ([FareRuleStruct](#) & *ioFareRule*)

Actor Constructor.

Definition at line 35 of file [FareParserHelper.cpp](#).

22.42.3 Member Function Documentation

22.42.3.1 void SIMFQT::FareParserHelper::storeFareId::operator() (unsigned int *iFareId*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 40 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), [SIMFQT::FareRuleStruct::_itSeconds](#), [SIMFQT::FareRuleStruct::clearAirlineCodeList\(\)](#), [SIMFQT::FareRuleStruct::clearClassCodeList\(\)](#), [SIMFQT::FareRuleStruct::setAirlineCode\(\)](#), [SIMFQT::FareRuleStruct::setClassCode\(\)](#), and [SIMFQT::FareRuleStruct::setFareId\(\)](#).

22.42.4 Member Data Documentation

22.42.4.1 [FareRuleStruct](#)& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

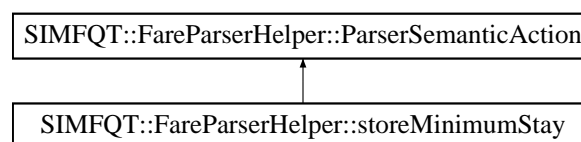
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.43 SIMFQT::FareParserHelper::storeMinimumStay Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeMinimumStay:



Public Member Functions

- [storeMinimumStay](#) ([FareRuleStruct](#) &)
- void [operator\(\)](#) (unsigned int, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.43.1 Detailed Description

Store the parsed minimum stay.

Definition at line 190 of file [FareParserHelper.hpp](#).

22.43.2 Constructor & Destructor Documentation

22.43.2.1 SIMFQT::FareParserHelper::storeMinimumStay::storeMinimumStay ([FareRuleStruct](#) & *ioFareRule*)

Actor Constructor.

Definition at line 343 of file [FareParserHelper.cpp](#).

22.43.3 Member Function Documentation

22.43.3.1 void SIMFQT::FareParserHelper::storeMinimumStay::operator() (unsigned int *iMinStay*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 348 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::setMinimumStay\(\)](#).

22.43.4 Member Data Documentation

22.43.4.1 **FareRuleStruct& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule** [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

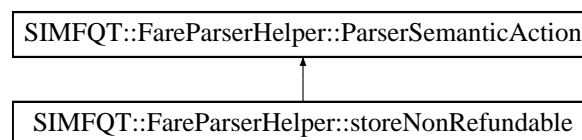
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.44 SIMFQT::FareParserHelper::storeNonRefundable Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeNonRefundable:



Public Member Functions

- [storeNonRefundable](#) ([FareRuleStruct](#) &)
- void [operator\(\)](#) (char, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.44.1 Detailed Description

Store the parsed refundable option

Definition at line 180 of file [FareParserHelper.hpp](#).

22.44.2 Constructor & Destructor Documentation

22.44.2.1 SIMFQT::FareParserHelper::storeNonRefundable::storeNonRefundable (FareRuleStruct & ioFareRule)

Actor Constructor.

Definition at line 318 of file [FareParserHelper.cpp](#).

22.44.3 Member Function Documentation

22.44.3.1 void SIMFQT::FareParserHelper::storeNonRefundable::operator() (char iNonRefundable, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 323 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::setNonRefundable\(\)](#).

22.44.4 Member Data Documentation

22.44.4.1 FareRuleStruct& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

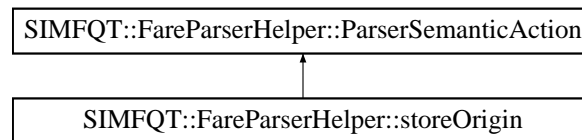
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.45 SIMFQT::FareParserHelper::storeOrigin Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeOrigin:



Public Member Functions

- [storeOrigin](#) ([FareRuleStruct](#) &)
- void [operator\(\)](#) (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.45.1 Detailed Description

Store the parsed origin.

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

22.45.2 Constructor & Destructor Documentation

22.45.2.1 SIMFQT::FareParserHelper::storeOrigin::storeOrigin ([FareRuleStruct](#) & *ioFareRule*)

Actor Constructor.

Definition at line 58 of file [FareParserHelper.cpp](#).

22.45.3 Member Function Documentation

22.45.3.1 void SIMFQT::FareParserHelper::storeOrigin::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 63 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::set-Origin\(\)](#).

22.45.4 Member Data Documentation

22.45.4.1 [FareRuleStruct](#)& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::Fare-](#)

[ParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

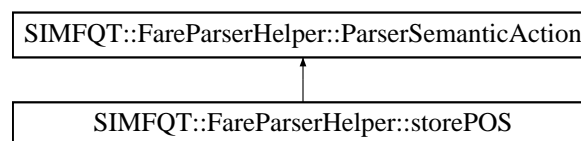
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.46 SIMFQT::FareParserHelper::storePOS Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storePOS:



Public Member Functions

- [storePOS](#) ([FareRuleStruct](#) &)
- [void operator\(\)](#) ([std::vector< char >](#), [boost::spirit::qi::unused_type](#), [boost::spirit::qi::unused_type](#)) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.46.1 Detailed Description

Store the parsed customer point_of_sale.

Definition at line 120 of file [FareParserHelper.hpp](#).

22.46.2 Constructor & Destructor Documentation

22.46.2.1 SIMFQT::FareParserHelper::storePOS::storePOS ([FareRuleStruct](#) & *ioFareRule*)

Actor Constructor.

Definition at line 185 of file [FareParserHelper.cpp](#).

22.46.3 Member Function Documentation

22.46.3.1 void SIMFQT::FareParserHelper::storePOS::operator() ([std::vector< char >](#) *iChar*, [boost::spirit::qi::unused_type](#), [boost::spirit::qi::unused_type](#)) const

Actor Function (functor).

Definition at line 190 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), [SIMFQT::FareRuleStruct::getDestination\(\)](#), [SIMFQT::FareRuleStruct::getOrigin\(\)](#), and [SIMFQT::FareRuleStruct::setPOS\(\)](#).

22.46.4 Member Data Documentation

22.46.4.1 FareRuleStruct& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

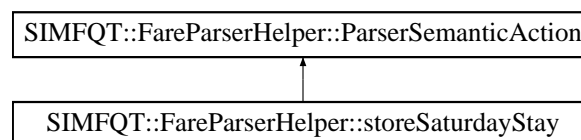
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.47 SIMFQT::FareParserHelper::storeSaturdayStay Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeSaturdayStay:



Public Member Functions

- [storeSaturdayStay](#) ([FareRuleStruct](#) &)
- void [operator\(\)](#) (char, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.47.1 Detailed Description

Store the parsed saturday night.

Definition at line 160 of file [FareParserHelper.hpp](#).

22.47.2 Constructor & Destructor Documentation

22.47.2.1 SIMFQT::FareParserHelper::storeSaturdayStay::storeSaturdayStay ([FareRuleStruct](#) & [ioFareRule](#))

Actor Constructor.

Definition at line 267 of file [FareParserHelper.cpp](#).

22.47.3 Member Function Documentation

22.47.3.1 void SIMFQT::FareParserHelper::storeSaturdayStay::operator() (char *iSaturdayStay*, boost::spirit::qi::unused_type , boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 272 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::set-SaturdayStay\(\)](#).

22.47.4 Member Data Documentation

22.47.4.1 FareRuleStruct& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

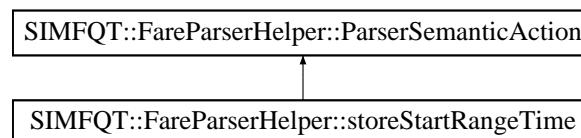
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.48 SIMFQT::FareParserHelper::storeStartRangeTime Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeStartRangeTime:



Public Member Functions

- [storeStartRangeTime](#) (FareRuleStruct &)
- void [operator\(\)](#) (boost::spirit::qi::unused_type, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.48.1 Detailed Description

Store the parsed start range time.

Definition at line 100 of file [FareParserHelper.hpp](#).

22.48.2 Constructor & Destructor Documentation

22.48.2.1 SIMFQT::FareParserHelper::storeStartRangeTime::storeStartRangeTime ([FareRuleStruct](#) & [ioFareRule](#))

Actor Constructor.

Definition at line 149 of file [FareParserHelper.cpp](#).

22.48.3 Member Function Documentation

22.48.3.1 void SIMFQT::FareParserHelper::storeStartRangeTime::operator() ([boost::spirit::qi::unused_type](#) , [boost::spirit::qi::unused_type](#) , [boost::spirit::qi::unused_type](#)) const

Actor Function (functor).

Definition at line 154 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), [SIMFQT::FareRuleStruct::_itSeconds](#), [SIMFQT::FareRuleStruct::calculateTime\(\)](#), and [SIMFQT::FareRuleStruct::setTimeRangeStart\(\)](#).

22.48.4 Member Data Documentation

22.48.4.1 [FareRuleStruct](#)& [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#) [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [SIMFQT::FareParserHelper::storeTripType::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvancePurchase::operator\(\)](#), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

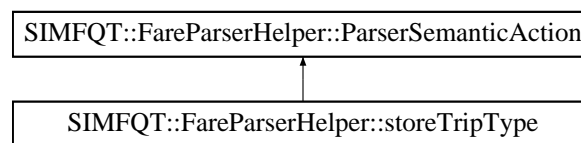
The documentation for this struct was generated from the following files:

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.49 SIMFQT::FareParserHelper::storeTripType Struct Reference

```
#include <simfqt/command/FareParserHelper.hpp>
```

Inheritance diagram for SIMFQT::FareParserHelper::storeTripType:



Public Member Functions

- [storeTripType](#) ([FareRuleStruct](#) &)
- [void operator\(\)](#) (std::vector< char >, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Public Attributes

- [FareRuleStruct](#) & [_fareRule](#)

22.49.1 Detailed Description

Store the parsed customer trip type.

Definition at line 69 of file [FareParserHelper.hpp](#).

22.49.2 Constructor & Destructor Documentation

22.49.2.1 SIMFQT::FareParserHelper::storeTripType::storeTripType ([FareRuleStruct](#) & *ioFareRule*)

Actor Constructor.

Definition at line 90 of file [FareParserHelper.cpp](#).

22.49.3 Member Function Documentation

22.49.3.1 void SIMFQT::FareParserHelper::storeTripType::operator() (std::vector< char > *iChar*, boost::spirit::qi::unused_type, boost::spirit::qi::unused_type) const

Actor Function (functor).

Definition at line 95 of file [FareParserHelper.cpp](#).

References [SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule](#), and [SIMFQT::FareRuleStruct::setTripType\(\)](#).

22.49.4 Member Data Documentation

22.49.4.1 [FareRuleStruct](#)& SIMFQT::FareParserHelper::ParserSemanticAction::_fareRule [inherited]

Actor Context.

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

Referenced by [SIMFQT::FareParserHelper::storeFareId::operator\(\)](#), [SIMFQT::FareParserHelper::storeOrigin::operator\(\)](#), [SIMFQT::FareParserHelper::storeDestination::operator\(\)](#), [operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeStart::operator\(\)](#), [SIMFQT::FareParserHelper::storeDateRangeEnd::operator\(\)](#), [SIMFQT::FareParserHelper::storeStartRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storeEndRangeTime::operator\(\)](#), [SIMFQT::FareParserHelper::storePOS::operator\(\)](#), [SIMFQT::FareParserHelper::storeCabinCode::operator\(\)](#), [SIMFQT::FareParserHelper::storeChannel::operator\(\)](#), [SIMFQT::FareParserHelper::storeAdvance](#)

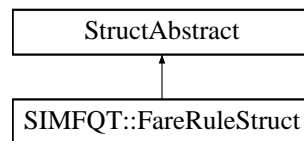
[Purchase::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeSaturdayStay::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeChangeFees::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeNonRefundable::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeMinimumStay::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeFare::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeAirlineCode::operator\(\)](#)(), [SIMFQT::FareParserHelper::storeClass::operator\(\)](#)(), and [SIMFQT::FareParserHelper::doEndFare::operator\(\)](#).

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

- [simfqt/command/FareParserHelper.hpp](#)
- [simfqt/command/FareParserHelper.cpp](#)

22.50 StructAbstract Class Reference

Inheritance diagram for StructAbstract:



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

- [simfqt/bom/FareRuleStruct.hpp](#)

23 File Documentation

23.1 [doc/local/authors.doc](#) File Reference

23.2 [doc/local/codingrules.doc](#) File Reference

23.3 [doc/local/copyright.doc](#) File Reference

23.4 [doc/local/documentation.doc](#) File Reference

23.5 [doc/local/features.doc](#) File Reference

23.6 [doc/local/help_wanted.doc](#) File Reference

23.7 [doc/local/howto_release.doc](#) File Reference

23.8 [doc/local/index.doc](#) File Reference

23.9 [doc/local/installation.doc](#) File Reference

23.10 [doc/local/linking.doc](#) File Reference

23.11 [doc/local/test.doc](#) File Reference

23.12 [doc/local/users_guide.doc](#) File Reference

23.13 [doc/local/verification.doc](#) File Reference

23.14 doc/tutorial/tutorial.doc File Reference

23.15 simfqt/basic/BasConst.cpp File Reference

```
#include <simfqt/basic/BasConst_General.hpp>
#include <simfqt/basic/BasConst_SIMFQT_Service.hpp>
```

Namespaces

- namespace [SIMFQT](#)

Variables

- const std::string [SIMFQT::DEFAULT_FARE_QUOTER_ID](#) = "IATA"

23.16 BasConst.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 #include <simfqt/basic/BasConst_General.hpp>
00005 #include <simfqt/basic/BasConst_SIMFQT_Service.hpp>
00006 >
00007 namespace SIMFQT {
00008
00010     const std::string DEFAULT_FARE_QUOTER_ID = "IATA";
00011
00012 }
```

23.17 simfqt/basic/BasConst_General.hpp File Reference

Namespaces

- namespace [SIMFQT](#)

23.18 BasConst_General.hpp

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

23.19 simfqt/basic/BasConst_SIMFQT_Service.hpp File Reference

```
#include <string>
```

Namespaces

- namespace [SIMFQT](#)

23.20 BasConst_SIMFQT_Service.hpp

```

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

```

23.21 simfqt/batches/simfqt_parseFareRules.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/bom/BookingRequestStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <simfqt/SIMFQT_Service.hpp>
#include <simfqt/config/simfqt-paths.hpp>

```

Typedefs

- typedef std::vector< std::string > [WordList_T](#)

Functions

- const std::string [K_SIMFQT_DEFAULT_LOG_FILENAME](#) ("simfqt_parseFareRules.log")
- const std::string [K_SIMFQT_DEFAULT_FARE_INPUT_FILENAME](#) (STDAIR_SAMPLE_DIR"/fare01.csv")
- template<class T >
std::ostream & [operator<<](#) (std::ostream &os, const std::vector< T > &v)
- int [readConfiguration](#) (int argc, char *argv[], bool &iolsBuiltin, stdair::Filename_T &ioFareInputFilename, std::string &ioLogFilename)
- int [main](#) (int argc, char *argv[])

Variables

- const bool [K_SIMFQT_DEFAULT_BUILT_IN_INPUT](#) = false
- const int [K_SIMFQT_EARLY_RETURN_STATUS](#) = 99

23.21.1 Typedef Documentation

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

Definition at line 24 of file [simfqt_parseFareRules.cpp](#).

23.21.2 Function Documentation

23.21.2.1 const std::string K_SIMFQT_DEFAULT_LOG_FILENAME ("simfqt_parseFareRules.log")

Default name and location for the log file.

Referenced by [readConfiguration\(\)](#).

23.21.2.2 const std::string K_SIMFQT_DEFAULT_FARE_INPUT_FILENAME (STDAIR_SAMPLE_DIR"/fare01.csv")

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

Referenced by [readConfiguration\(\)](#).

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

Definition at line 44 of file [simfqt_parseFareRules.cpp](#).

23.21.2.4 int readConfiguration (int argc, char * argv[], bool & iolsBuiltin, stdair::Filename_T & ioFareInputFilename, std::string & ioLogFilename)

Read and parse the command line options.

Definition at line 51 of file [simfqt_parseFareRules.cpp](#).

References [K_SIMFQT_DEFAULT_BUILT_IN_INPUT](#), [K_SIMFQT_DEFAULT_FARE_INPUT_FILENAME\(\)](#), [K_SIMFQT_DEFAULT_LOG_FILENAME\(\)](#), [K_SIMFQT_EARLY_RETURN_STATUS](#), [PACKAGE_NAME](#), [PACKAGE_VERSION](#), and [PREFIXDIR](#).

Referenced by [main\(\)](#).

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

Definition at line 154 of file [simfqt_parseFareRules.cpp](#).

References [SIMFQT::SIMFQT_Service::buildBookingRequest\(\)](#), [SIMFQT::SIMFQT_Service::buildSampleBom\(\)](#), [SIMFQT::SIMFQT_Service::buildSampleTravelSolutions\(\)](#), [SIMFQT::SIMFQT_Service::csvDisplay\(\)](#), [K_SIMFQT_EARLY_RETURN_STATUS](#), [SIMFQT::SIMFQT_Service::parseAndLoad\(\)](#), [SIMFQT::SIMFQT_Service::quotePrices\(\)](#), and [readConfiguration\(\)](#).

23.21.3 Variable Documentation

23.21.3.1 const bool K_SIMFQT_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 37 of file [simfqt_parseFareRules.cpp](#).

Referenced by [readConfiguration\(\)](#).

23.21.3.2 const int K_SIMFQT_EARLY_RETURN_STATUS = 99

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

Definition at line 40 of file [simfqt_parseFareRules.cpp](#).

Referenced by [main\(\)](#), and [readConfiguration\(\)](#).

23.22 simfqt_parseFareRules.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/bom/BookingRequestStruct.hpp>
00018 #include <stdair/service/Logger.hpp>
00019 // Simfqt
00020 #include <simfqt/SIMFQT_Service.hpp>
00021 #include <simfqt/config/simfqt-paths.hpp>
00022
00023 // ////////// Type definitions //////////
00024 typedef std::vector<std::string> WordList_T;
00025
00026
00027 // ////////// Constants //////////
00028 const std::string K_SIMFQT_DEFAULT_LOG_FILENAME ("
00029     simfqt_parseFareRules.log");
00030
00031
00032 const std::string K_SIMFQT_DEFAULT_FARE_INPUT_FILENAME
00033     (STDAIR_SAMPLE_DIR
00034          "/fare01.csv");
00035
00036 const bool K_SIMFQT_DEFAULT_BUILT_IN_INPUT =
00037     false;
00038
00039 const int K_SIMFQT_EARLY_RETURN_STATUS = 99;
00040
00041
00042 // ////////// Parsing of Options & Configuration //////////
00043 // A helper function to simplify the main part.
00044 template<class T> std::ostream& operator<< (std::ostream& os,
00045     const std::vector<T>& v) {
00046     std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00047     return os;
00048 }
00049
00050 int readConfiguration (int argc, char* argv[], bool&
00051     ioIsBuiltin,
00052     stdair::Filename_T& ioFareInputFilename,
00053     std::string& ioLogFilename) {
00054
00055     // Default for the built-in input
00056     ioIsBuiltin = K_SIMFQT_DEFAULT_BUILT_IN_INPUT;
00057
00058     // Declare a group of options that will be allowed only on command line
00059     boost::program_options::options_description generic ("Generic options");
00060     generic.add_options()
00061         ("prefix", "print installation prefix")
00062         ("version,v", "print version string")
00063         ("help,h", "produce help message");
00064
00065     // Declare a group of options that will be allowed both on command
00066     // line and in config file
00067     boost::program_options::options_description config ("Configuration");
00068     config.add_options()
00069         ("builtin,b",
00070             "The sample BOM tree can be either built-in or parsed from an input file.
00071             That latter must then be given with the -f/--fare option")
00072         ("fare,f",
00073             boost::program_options::value< std::string >(&ioFareInputFilename)->
00074             default_value(K_SIMFQT_DEFAULT_FARE_INPUT_FILENAME
00075                 ),
00076             "(CSV) input file for the fare rules")
00077         ("log,l",
00078             boost::program_options::value< std::string >(&ioLogFilename)->
00079             default_value(K_SIMFQT_DEFAULT_LOG_FILENAME),
00080             "Filename for the logs")
00081         ;
00082
00083     // Hidden options, will be allowed both on command line and
00084     // in config file, but will not be shown to the user.
00085     boost::program_options::options_description hidden ("Hidden options");
00086     hidden.add_options()
00087         ("copyright",

```



```

00084     boost::program_options::value< std::vector<std::string> >(),
00085     "Show the copyright (license)");
00086
00087     boost::program_options::options_description cmdline_options;
00088     cmdline_options.add(generic).add(config).add(hidden);
00089
00090     boost::program_options::options_description config_file_options;
00091     config_file_options.add(config).add(hidden);
00092
00093     boost::program_options::options_description visible ("Allowed options");
00094     visible.add(generic).add(config);
00095
00096     boost::program_options::positional_options_description p;
00097     p.add ("copyright", -1);
00098
00099     boost::program_options::variables_map vm;
00100     boost::program_options::
00101         store (boost::program_options::command_line_parser (argc, argv).
00102             options (cmdline_options).positional(p).run(), vm);
00103
00104     std::ifstream ifs ("simfqt.cfg");
00105     boost::program_options::store (parse_config_file (ifs, config_file_options),
00106         vm);
00107     boost::program_options::notify (vm); if (vm.count ("help")) {
00108         std::cout << visible << std::endl;
00109         return K_SIMFQT_EARLY_RETURN_STATUS;
00110     }
00111
00112     if (vm.count ("version")) {
00113         std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION
00114         << std::endl;
00115         return K_SIMFQT_EARLY_RETURN_STATUS;
00116     }
00117
00118     if (vm.count ("prefix")) {
00119         std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00120         return K_SIMFQT_EARLY_RETURN_STATUS;
00121     }
00122
00123     if (vm.count ("builtin")) {
00124         ioIsBuiltin = true;
00125     }
00126     const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00127     std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00128
00129     if (ioIsBuiltin == false) {
00130         // The BOM tree should be built from parsing a fare (and O&D) file
00131         if (vm.count ("fare")) {
00132             ioFareInputFilename = vm["fare"].as< std::string >();
00133             std::cout << "Input fare filename is: " << ioFareInputFilename
00134                 << std::endl;
00135         } else {
00136             // The built-in option is not selected. However, no fare file
00137             // is specified
00138             std::cerr << "Either one among the -b/--builtin and -f/--fare "
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 // ////////////////////////////////// M A I N //////////////////////////////////
00152 int main (int argc, char* argv[]) {
00153
00154     // State whether the BOM tree should be built-in or parsed from an input file
00155     bool isBuiltin;
00156
00157     // Fare input filename
00158     stdair::Filename_T lFareInputFilename;
00159
00160     // Output log File
00161     stdair::Filename_T lLogFilename;
00162
00163     // Call the command-line option parser
00164     const int lOptionParserStatus =
00165         readConfiguration (argc, argv, isBuiltin,
00166             lFareInputFilename, lLogFilename);
00167
00168

```

```

00169     if (lOptionParserStatus == K_SIMFQT_EARLY_RETURN_STATUS
00170     ) {
00171         return 0;
00172     }
00173     // Set the log parameters
00174     std::ofstream logOutputFile;
00175     // Open and clean the log outputfile
00176     logOutputFile.open (lLogFilename.c_str());
00177     logOutputFile.clear();
00178
00179     // Initialise the Simfqt service object
00180     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00181
00182     SIMFQT::SIMFQT_Service simfqtService (lLogParams);
00183
00184     // DEBUG
00185     STDAIR_LOG_DEBUG ("Welcome to Simfqt");
00186
00187     // Build a default sample list of travel solutions
00188     stdair::TravelSolutionList_T lTravelSolutionList;
00189     simfqtService.buildSampleTravelSolutions (
00190     lTravelSolutionList);
00191
00192     // Build a default booking request
00193     stdair::BookingRequestStruct lBookingRequest =
00194     simfqtService.buildBookingRequest();
00195
00196     // Check wether or not a (CSV) input file should be read
00197     if (isBuiltin == true) {
00198         // Build the default sample BOM tree (filled with fares) for Simfqt
00199         simfqtService.buildSampleBom();
00200     } else {
00201         // Build the BOM tree from parsing a fare file
00202         SIMFQT::FareFilePath lFareFilePath (lFareInputFilename)
00203     ;
00204         simfqtService.parseAndLoad (lFareFilePath);
00205     }
00206
00207     // DEBUG: Display the travel solutions
00208     const std::string& lTSCSVDump =
00209     simfqtService.csvDisplay (lTravelSolutionList);
00210     STDAIR_LOG_DEBUG (lTSCSVDump);
00211
00212     // FareQuote the sample list of travel solutions
00213     simfqtService.quotePrices (lBookingRequest, lTravelSolutionList);
00214
00215     // DEBUG: Display the whole BOM tree
00216     const std::string& lBOMCSVDump = simfqtService.csvDisplay();
00217     STDAIR_LOG_DEBUG ("BOM tree: " << lBOMCSVDump);
00218
00219     // DEBUG: Display the travel solutions
00220     const std::string& lTSCSVDumpEnd
00221     = simfqtService.csvDisplay (lTravelSolutionList);
00222     STDAIR_LOG_DEBUG (lTSCSVDumpEnd);
00223
00224     // Close the Log outputFile
00225     logOutputFile.close();
00226
00227     /*
00228     Note: as that program is not intended to be run on a server in
00229     production, it is better not to catch the exceptions. When it
00230     happens (that an exception is throwned), that way we get the
00231     call stack.
00232     */
00233     return 0;
00234 }
00235
00236 #include <cassert>

```

23.23 simfqt/bom/FareRuleStruct.cpp File Reference

```
#include <cassert>
```

```
#include <sstream>
#include <vector>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/service/Logger.hpp>
#include <simfqt/bom/FareRuleStruct.hpp>
```

Namespaces

- namespace **SIMFQT**

23.24 FareRuleStruct.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #include <vector>
00008 // StdAir
00009 #include <stdair/basic/BasConst_General.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 // SIMFQT
00012 #include <simfqt/bom/FareRuleStruct.hpp>
00013
00014 namespace SIMFQT {
00015
00016 // //////////////////////////////////////
00017 FareRuleStruct::FareRuleStruct ()
00018     :_fareId(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     _cabinCode(""),
00026     _pos(""),
00027     _advancePurchase(0),
00028     _saturdayStay("T"),
00029     _changeFees("T"),
00030     _nonRefundable("T"),
00031     _minimumStay(0),
00032     _fare(0),
00033     _airlineCode(""),
00034     _classCode("") {
00035
00036 }
00037
00038 // //////////////////////////////////////
00039 stdair::Date_T FareRuleStruct::calculateDate()
00040 const {
00041     _itYear.check(); _itMonth.check(); _itDay.check();
00042     return stdair::Date_T (_itYear._value, _itMonth._value,
00043         _itDay._value);
00044 }
00045 // //////////////////////////////////////
00046 stdair::Duration_T FareRuleStruct::calculateTime
00047 () const {
00048     _itHours.check(); _itMinutes.check(); _itSeconds
00049     .check();
00050     return boost::posix_time::hours (_itHours._value)
00051         + boost::posix_time::minutes (_itMinutes._value)
00052         + boost::posix_time::seconds (_itSeconds._value);
00053 }
00054 // //////////////////////////////////////
00055 const std::string FareRuleStruct::describe () const {
00056     std::ostringstream oStr;
00057     oStr << "FareRule: " << _fareId << ", ";
00058
00059     oStr << _origin << "-" << _destination << " ("
00060         << _pos << ")", " << _channel << ", [";
00061     oStr << _dateRangeStart << "/" << _dateRangeEnd << "]" - ["
00062         << boost::posix_time::to_simple_string (_timeRangeStart) << "/"
```

```

00063         << boost::posix_time::to_simple_string (_timeRangeEnd) << "], ";
00064
00065         oStr << _cabinCode << ", " << _fare << " EUR, ";
00066         oStr << _tripType << ", " << _saturdayStay << ", "
00067         << _changeFees << ", " << _nonRefundable << ", "
00068         << _advancePurchase << ", " << _minimumStay << ", ";
00069
00070         // Sanity check
00071         assert (_airlineCodeList.size() == _classCodeList.size());
00072
00073         // Browse the airline and class pathes
00074         unsigned short idx = 0;
00075         stdair::ClassList_StringList_T::const_iterator itClass =
00076             _classCodeList.begin();
00077         for (stdair::AirlineCodeList_T::const_iterator itAirline =
00078             _airlineCodeList.begin();
00079             itAirline != _airlineCodeList.end(); ++itAirline, ++itClass, ++idx) {
00080             if (idx != 0) {
00081                 oStr << " - ";
00082             }
00083             const stdair::AirlineCode_T lAirlineCode = *itAirline;
00084             const stdair::ClassCode_T lClassCode = *itClass;
00085             oStr << lAirlineCode << " / " << lClassCode;
00086         }
00087
00088         return oStr.str();
00089     }
00090
00091 }
00092

```

23.25 simfqt/bom/FareRuleStruct.hpp File Reference

```

#include <string>
#include <vector>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/basic/BasParserHelperTypes.hpp>
#include <simfqt/SIMFQT_Types.hpp>

```

Classes

- struct [SIMFQT::FareRuleStruct](#)

Namespaces

- namespace [SIMFQT](#)

23.26 FareRuleStruct.hpp

```

00001 #ifndef __SIMFQT_BOM_FARERULESTRUCT_HPP
00002 #define __SIMFQT_BOM_FARERULESTRUCT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // StdAir
00011 #include <stdair/stdair_demand_types.hpp>
00012 #include <stdair/stdair_inventory_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/basic/BasParserHelperTypes.hpp>
00015 // SIMFQT
00016 #include <simfqt/SIMFQT_Types.hpp>
00017
00018 namespace SIMFQT {
00019

```

```
00021 struct FareRuleStruct : public stdair::StructAbstract {
00022 public:
00023
00025     FareRuleStruct ();
00026
00027 public:
00028     // ////////// Getters //////////
00030     SIMFQT::FareQuoteID_T getFareID () const {
00031         return _fareId;
00032     }
00033
00035     stdair::AirportCode_T getOrigin () const {
00036         return _origin;
00037     }
00038
00040     stdair::AirportCode_T getDestination () const {
00041         return _destination;
00042     }
00043
00045     stdair::TripType_T getTripType () const {
00046         return _tripType;
00047     }
00048
00050     stdair::Date_T getDateRangeStart () const {
00051         return _dateRangeStart;
00052     }
00053
00055     stdair::Date_T getDateRangeEnd () const {
00056         return _dateRangeEnd;
00057     }
00058
00060     stdair::Duration_T getTimeRangeStart () const {
00061         return _timeRangeStart;
00062     }
00063
00065     stdair::Duration_T getTimeRangeEnd () const {
00066         return _timeRangeEnd;
00067     }
00068
00070     stdair::CabinCode_T getCabinCode () const {
00071         return _cabinCode;
00072     }
00073
00075     const stdair::CityCode_T getPOS () const {
00076         return _pos;
00077     }
00078
00080     stdair::ChannelLabel_T getChannel () const {
00081         return _channel;
00082     }
00083
00085     stdair::DayDuration_T getAdvancePurchase () const {
00086         return _advancePurchase;
00087     }
00088
00090     stdair::SaturdayStay_T getSaturdayStay () const {
00091         return _saturdayStay;
00092     }
00093
00095     stdair::ChangeFees_T getChangeFees () const {
00096         return _changeFees;
00097     }
00098
00100     stdair::NonRefundable_T getNonRefundable () const {
00101         return _nonRefundable;
00102     }
00103
00105     stdair::DayDuration_T getMinimumStay () const {
00106         return _minimumStay;
00107     }
00108
00110     stdair::PriceValue_T getFare () const {
00111         return _fare;
00112     }
00113
00115     stdair::AirlineCode_T getAirlineCode () const {
00116         return _airlineCode;
00117     }
00118
00120     stdair::ClassCode_T getClassCode () const {
00121         return _classCode;
00122     }
00123
00125     const unsigned int getAirlineListSize () const {
00126         return _airlineCodeList.size();
00127     }
00128
```

```

00130     const unsigned int getClassCodeListSize () const {
00131         return _classCodeList.size();
00132     }
00133
00135     stdair::AirlineCodeList_T getAirlineList () const {
00136         return _airlineCodeList;
00137     }
00138
00140     stdair::ClassList_StringList_T getClassCodeList () const {
00141         return _classCodeList;
00142     }
00143
00144 public:
00145     // ////////// Display support methods //////////
00147     stdair::Date_T calculateDate() const;
00148
00150     stdair::Duration_T calculateTime() const;
00151
00153     const std::string describe() const;
00154
00155 public:
00156     // ////////// Setters //////////
00158     void setFareID (const SIMFQT::FareQuoteID_T&
00159 iFareQuoteID) {
00160         _fareId = iFareQuoteID;
00161     }
00163     void setOrigin (const stdair::AirportCode_T& iOrigin) {
00164         _origin = iOrigin;
00165     }
00166
00168     void setDestination (const stdair::AirportCode_T&
00169 iDestination) {
00170         _destination = iDestination;
00171     }
00173     void setTripType (const stdair::TripType_T& iTripType) {
00174         _tripType = iTripType;
00175     }
00176
00178     void setDateRangeStart (const stdair::Date_T&
00179 iDateRangeStart) {
00180         _dateRangeStart = iDateRangeStart;
00181     }
00183     void setDateRangeEnd (const stdair::Date_T& iDateRangeEnd) {
00184         _dateRangeEnd = iDateRangeEnd;
00185     }
00186
00188     void setTimeRangeStart (const stdair::Duration_T&
00189 iTimeRangeStart) {
00190         _timeRangeStart = iTimeRangeStart;
00191     }
00193     void setTimeRangeEnd (const stdair::Duration_T&
00194 iTimeRangeEnd) {
00195         _timeRangeEnd = iTimeRangeEnd;
00196     }
00198     void setCabinCode (const stdair::CabinCode_T& iCabinCode) {
00199         _cabinCode = iCabinCode;
00200     }
00201
00203     void setPOS (const stdair::CityCode_T& iPOS) {
00204         _pos = iPOS;
00205     }
00206
00208     void setChannel (const stdair::ChannelLabel_T& iChannel) {
00209         _channel = iChannel;
00210     }
00211
00213     void setAdvancePurchase (const stdair::DayDuration_T&
00214 iAdvancePurchase) {
00215         _advancePurchase = iAdvancePurchase;
00216     }
00218     void setSaturdayStay (const stdair::SaturdayStay_T&
00219 iSaturdayStay) {
00220         _saturdayStay = iSaturdayStay;
00221     }
00223     void setChangeFees (const stdair::ChangeFees_T& iChangeFees) {
00224         _changeFees = iChangeFees;
00225     }
00226
00228     void setNonRefundable (const stdair::NonRefundable_T&
00229 iNonRefundable) {

```

```
00229     _nonRefundable = iNonRefundable;
00230 }
00231
00233 void setMinimumStay (const stdair::DayDuration_T&
iMinimumStay) {
00234     _minimumStay = iMinimumStay;
00235 }
00236
00238 void setFare (const stdair::PriceValue_T& iFare) {
00239     _fare = iFare;
00240 }
00241
00243 void setAirlineCode (const stdair::AirlineCode_T&
iAirlineCode) {
00244     _airlineCode = iAirlineCode;
00245 }
00246
00248 void setClassCode (const stdair::ClassCode_T& iClassCode) {
00249     _classCode = iClassCode;
00250 }
00251
00253 void clearAirlineCodeList () {
00254     _airlineCodeList.clear();
00255 }
00256
00258 void clearClassCodeList () {
00259     _classCodeList.clear();
00260 }
00261
00263 void addAirlineCode (const stdair::AirlineCode_T&
iAirlineCode) {
00264     _airlineCodeList.push_back (iAirlineCode);
00265 }
00266
00268 void addClassCode (const stdair::ClassCode_T& iClassCode) {
00269     _classCodeList.push_back (iClassCode);
00270 }
00271
00272 public:
00273     // ////////////////////////////////// Attributes //////////////////////////////////
00275     stdair::year_t _itYear;
00276     stdair::month_t _itMonth;
00277     stdair::day_t _itDay;
00278
00280     stdair::hour_t _itHours;
00281     stdair::minute_t _itMinutes;
00282     stdair::second_t _itSeconds;
00283
00284 private:
00285     // ////////////////////////////////// Attributes //////////////////////////////////
00287     SIMFQT::FareQuoteID_T _fareId;
00288
00290     stdair::AirportCode_T _origin;
00291
00293     stdair::AirportCode_T _destination;
00294
00296     stdair::TripType_T _tripType;
00297
00299     stdair::Date_T _dateRangeStart;
00300
00302     stdair::Date_T _dateRangeEnd;
00303
00305     stdair::Duration_T _timeRangeStart;
00306
00308     stdair::Duration_T _timeRangeEnd;
00309
00311     stdair::CabinCode_T _cabinCode;
00312
00314     stdair::CityCode_T _pos;
00315
00317     stdair::ChannelLabel_T _channel;
00318
00320     stdair::DayDuration_T _advancePurchase;
00321
00323     stdair::SaturdayStay_T _saturdayStay;
00324
00326     stdair::ChangeFees_T _changeFees;
00327
00329     stdair::NonRefundable_T _nonRefundable;
00330
00332     stdair::DayDuration_T _minimumStay;
00333
00335     stdair::PriceValue_T _fare;
00336
00338     stdair::AirlineCode_T _airlineCode;
00339
00341     stdair::ClassCode_T _classCode;
```

```

00342
00345     stdair::AirlineCodeList_T _airlineCodeList;
00346
00349     stdair::ClassList_StringList_T _classCodeList;
00350
00351 };
00352
00353 }
00354 #endif // __SIMFQT_BOM_FARERULESTRUCT_HPP

```

23.27 simfqt/command/FareParser.cpp File Reference

```

#include <cassert>
#include <string>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/service/Logger.hpp>
#include <simfqt/command/FareParserHelper.hpp>
#include <simfqt/command/FareParser.hpp>

```

Namespaces

- namespace [SIMFQT](#)

23.28 FareParser.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <string>
00007 // StdAir
00008 #include <stdair/basic/BasFileMgr.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 // AirSched
00011 #include <simfqt/command/FareParserHelper.hpp>
00012 #include <simfqt/command/FareParser.hpp>
00013
00014 namespace SIMFQT {
00015
00016 // //////////////////////////////////////
00017 void FareParser::fareRuleGeneration (const
FareFilePath& iFareFilename,
                                stdair::BomRoot& ioBomRoot) {
00018
00019     const stdair::Filename_T lFilename = iFareFilename.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 fare input file, '" << lFilename
00026             << "', can not be retrieved on the file-system");
00027         throw FareInputFileNotFoundExpection ("The
fare input file '" + lFilename
00028             + "' does not exist or can not "
00029             "be read");
00030     }
00031
00032     // Initialise the fare file parser.
00033     FareRuleFileParser lFareRuleFileParser (ioBomRoot,
lFilename);
00034
00035     // Parse the CSV-formatted fare input file and generate the
00036     // corresponding fare rules.
00037     lFareRuleFileParser.generateFareRules ();
00038
00039 }
00040
00041 }
00042

```


23.29 simfqt/command/FareParser.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/command/CmdAbstract.hpp>
#include <simfqt/SIMFQT_Types.hpp>
```

Classes

- class [SIMFQT::FareParser](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [SIMFQT](#)

23.30 FareParser.hpp

```
00001 #ifndef __SIMFQT_CMD_FAREPARSER_HPP
00002 #define __SIMFQT_CMD_FAREPARSER_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 // SimFQT
00013 #include <simfqt/SIMFQT_Types.hpp>
00014
00015 // Forward declarations.
00016 namespace stdair {
00017     class BomRoot;
00018 }
00019
00020 namespace SIMFQT {
00021
00022     class FareParser : public stdair::CmdAbstract {
00023     public:
00024         static void fareRuleGeneration (const FareFilePath
00030         &, stdair::BomRoot&);
00031     };
00032 }
00033 #endif // __SIMFQT_CMD_FAREPARSER_HPP
```

23.31 simfqt/command/FareParserHelper.cpp File Reference

```
#include <cassert>
#include <vector>
#include <fstream>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/basic/BasParserTypes.hpp>
#include <simfqt/command/FareParserHelper.hpp>
#include <simfqt/command/FareRuleGenerator.hpp>
```

Classes

- struct `SIMFQT::FareParserHelper::FareRuleParser< Iterator >`

Namespaces

- namespace `SIMFQT`
- namespace `SIMFQT::FareParserHelper`

Variables

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

23.32 FareParserHelper.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <vector>
00007 #include <fstream>
00008 // StdAir
00009 #include <stdair/basic/BasFileMgr.hpp>
00010 #include <stdair/bom/BomRoot.hpp>
00011 #include <stdair/service/Logger.hpp>
00012 // #define BOOST_SPIRIT_DEBUG
00013 #include <stdair/basic/BasParserTypes.hpp>
00014 // SIMFQT
00015 #include <simfqt/command/FareParserHelper.hpp>
00016 #include <simfqt/command/FareRuleGenerator.hpp>
00017
00018
00019 namespace SIMFQT {
00020     namespace FareParserHelper {
00021
00022         //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00023         // Semantic actions
00024         //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00025
00026         ParserSemanticAction::
00027         ParserSemanticAction (FareRuleStruct&
00028         ioFareRule)
00029         : _fareRule (ioFareRule) {
00030
00031         }
00032
00033         //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00034         storeFareId::
00035         storeFareId (FareRuleStruct& ioFareRule)
00036         : ParserSemanticAction (ioFareRule) {
00037
00038         }
00039
00040         //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00041         void storeFareId::operator() (unsigned int iFareId,
00042         boost::spirit::qi::unused_type,
00043         boost::spirit::qi::unused_type) const {
00044             _fareRule.setFareID (iFareId);
00045         }
00046     }
00047 }

```

```

00045         // DEBUG
00046         //STDAIR_LOG_DEBUG ( "Fare Id: " << _fareRule.getFareID ());
00047         const stdair::AirlineCode_T lEmptyAirlineCode ("");
00048         _fareRule.setAirlineCode(lEmptyAirlineCode);
00049         _fareRule.clearAirlineCodeList();
00050         const stdair::ClassCode_T lEmptyClassCode ("");
00051         _fareRule.setClassCode(lEmptyClassCode);
00052         _fareRule.clearClassCodeList();
00053         _fareRule._itSeconds = 0;
00054     }
00055
00056     // //////////////////////////////////////
00057     storeOrigin ::
00058     storeOrigin (FareRuleStruct& ioFareRule)
00059         : ParserSemanticAction (ioFareRule) {
00060     }
00061
00062     // //////////////////////////////////////
00063     void storeOrigin::operator() (std::vector<char>
iChar,
00064                                     boost::spirit::qi::unused_type,
00065                                     boost::spirit::qi::unused_type) const {
00066         const stdair::AirportCode_T lOrigin (iChar.begin(), iChar.end());
00067         _fareRule.setOrigin (lOrigin);
00068         // DEBUG
00069         //STDAIR_LOG_DEBUG ( "Origin: " << _fareRule.getOrigin ());
00070     }
00071
00072     // //////////////////////////////////////
00073     storeDestination ::
00074     storeDestination (FareRuleStruct&
ioFareRule)
00075         : ParserSemanticAction (ioFareRule) {
00076     }
00077
00078     // //////////////////////////////////////
00079     void storeDestination::operator() (
std::vector<char> iChar,
00080                                     boost::spirit::qi::unused_type,
00081                                     boost::spirit::qi::unused_type) const {
00082         const stdair::AirportCode_T lDestination (iChar.begin(), iChar.end());
00083         _fareRule.setDestination (lDestination);
00084         // DEBUG
00085         //STDAIR_LOG_DEBUG ( "Destination: " << _fareRule.getDestination ());
00086     }
00087
00088     // //////////////////////////////////////
00089     storeTripType ::
00090     storeTripType (FareRuleStruct& ioFareRule)
00091         : ParserSemanticAction (ioFareRule) {
00092     }
00093
00094     // //////////////////////////////////////
00095     void storeTripType::operator() (std::vector<char>
iChar,
00096                                     boost::spirit::qi::unused_type,
00097                                     boost::spirit::qi::unused_type) const {
00098         const stdair::TripType_T lTripType (iChar.begin(), iChar.end());
00099         if (lTripType == "OW" || lTripType == "RT") {
00100             _fareRule.setTripType (lTripType);
00101         } else {
00102             // ERROR
00103             STDAIR_LOG_ERROR ("Invalid trip type " << lTripType);
00104         }
00105         // DEBUG
00106         //STDAIR_LOG_DEBUG ("TripType: " << _fareRule.getTripType ());
00107     }
00108
00109     // //////////////////////////////////////
00110     storeDateRangeStart ::
00111     storeDateRangeStart (FareRuleStruct&
ioFareRule)
00112         : ParserSemanticAction (ioFareRule) {
00113     }
00114
00115     // //////////////////////////////////////
00116     void storeDateRangeStart::operator() (
boost::spirit::qi::unused_type,
00117                                     boost::spirit::qi::unused_type,
00118                                     boost::spirit::qi::unused_type) const
00119     {
00120         const stdair::Date_T& lDateStart = _fareRule.calculateDate
();
00121         _fareRule.setDateRangeStart (lDateStart);
00122         // DEBUG
00123         //STDAIR_LOG_DEBUG ("Date Range Start: " << _fareRule.getDateRangeStart

```

```

    );
00124     }
00125
00126     // //////////////////////////////////////
00127     storeDateRangeEnd::
00128     storeDateRangeEnd(FareRuleStruct&
ioFareRule)
00129     : ParserSemanticAction (ioFareRule) {
00130     }
00131
00132     // //////////////////////////////////////
00133     void storeDateRangeEnd::operator() (
boost::spirit::qi::unused_type,
00134                                     boost::spirit::qi::unused_type,
00135                                     boost::spirit::qi::unused_type) const {
00136         const stdair::Date_T& lDateEnd = _fareRule.calculateDate
    );
00137         // As a Boost date period (DatePeriod_T) defines the last day of
00138         // the period to be end-date - one day, we have to add one day to that
00139         // end date before.
00140         const stdair::DateOffset_T oneDay (1);
00141         const stdair::Date_T lBoostDateEnd = lDateEnd + oneDay;
00142         _fareRule.setDateRangeEnd (lBoostDateEnd);
00143         // DEBUG
00144         //STDAIR_LOG_DEBUG ("Date Range End: " << _fareRule.getDateRangeEnd ());
00145     }
00146
00147     // //////////////////////////////////////
00148     storeStartRangeTime::
00149     storeStartRangeTime (FareRuleStruct&
ioFareRule)
00150     : ParserSemanticAction (ioFareRule) {
00151     }
00152
00153     // //////////////////////////////////////
00154     void storeStartRangeTime::operator() (
boost::spirit::qi::unused_type,
00155                                     boost::spirit::qi::unused_type,
00156                                     boost::spirit::qi::unused_type) const
    {
00157         const stdair::Duration_T& lTimeStart = _fareRule.calculateTime
    );
00158         _fareRule.setTimeRangeStart (lTimeStart);
00159         // DEBUG
00160         //STDAIR_LOG_DEBUG ("Time Range Start: " << _fareRule.getTimeRangeStart
    );
00161         // Reset the number of seconds
00162         _fareRule._itSeconds = 0;
00163     }
00164
00165     // //////////////////////////////////////
00166     storeEndRangeTime::
00167     storeEndRangeTime (FareRuleStruct&
ioFareRule)
00168     : ParserSemanticAction (ioFareRule) {
00169     }
00170
00171     // //////////////////////////////////////
00172     void storeEndRangeTime::operator() (
boost::spirit::qi::unused_type,
00173                                     boost::spirit::qi::unused_type,
00174                                     boost::spirit::qi::unused_type) const {
00175         const stdair::Duration_T& lTimeEnd = _fareRule.calculateTime
    );
00176         _fareRule.setTimeRangeEnd (lTimeEnd);
00177         // DEBUG
00178         //STDAIR_LOG_DEBUG ("Time Range End: " << _fareRule.getTimeRangeEnd ());
00179         // Reset the number of seconds
00180         _fareRule._itSeconds = 0;
00181     }
00182
00183     // //////////////////////////////////////
00184     storePOS ::
00185     storePOS (FareRuleStruct& ioFareRule)
00186     : ParserSemanticAction (ioFareRule) {
00187     }
00188
00189     // //////////////////////////////////////
00190     void storePOS::operator() (std::vector<char> iChar,
boost::spirit::qi::unused_type,
00191                                     boost::spirit::qi::unused_type) const {
00192         const stdair::CityCode_T lPOS (iChar.begin(), iChar.end());
00193         if (lPOS == _fareRule.getOrigin() || lPOS == _fareRule
.getDestination()) {
00194             _fareRule.setPOS (lPOS);
00195         } else if (lPOS == "ROW") {
00196             const stdair::CityCode_T lPOSROW ("ROW");
00197

```

```

00198     _fareRule.setPOS (lPOSROW);
00199 } else {
00200     // ERROR
00201     STDAIR_LOG_ERROR ("Invalid point of sale " << lPOS);
00202 }
00203 // DEBUG
00204 //STDAIR_LOG_DEBUG ("POS: " << _fareRule.getPOS ());
00205 }
00206
00207 // //////////////////////////////////////
00208 storeCabinCode ::
00209 storeCabinCode (FareRuleStruct& ioFareRule)
00210 : ParserSemanticAction (ioFareRule) {
00211 }
00212
00213 // //////////////////////////////////////
00214 void storeCabinCode::operator() (char iChar,
00215                                 boost::spirit::qi::unused_type,
00216                                 boost::spirit::qi::unused_type) const {
00217     std::ostringstream ostr;
00218     ostr << iChar;
00219     const std::string cabinCodeStr = ostr.str();
00220     const stdair::CabinCode_T& lCabinCode (cabinCodeStr);
00221     _fareRule.setCabinCode (lCabinCode);
00222
00223     // DEBUG
00224     //STDAIR_LOG_DEBUG ("Cabin Code: " << _fareRule.getCabinCode ());
00225
00226 }
00227
00228 // //////////////////////////////////////
00229 storeChannel ::
00230 storeChannel (FareRuleStruct& ioFareRule)
00231 : ParserSemanticAction (ioFareRule) {
00232 }
00233
00234 // //////////////////////////////////////
00235 void storeChannel::operator() (std::vector<char>
iChar,
00236                               boost::spirit::qi::unused_type,
00237                               boost::spirit::qi::unused_type) const {
00238     const stdair::ChannelLabel_T lChannel (iChar.begin(), iChar.end());
00239     if (lChannel != "IN" && lChannel != "IF"
00240         && lChannel != "DN" && lChannel != "DF") {
00241         // ERROR
00242         STDAIR_LOG_ERROR ("Invalid channel " << lChannel);
00243     }
00244     _fareRule.setChannel (lChannel);
00245     // DEBUG
00246     //STDAIR_LOG_DEBUG ("Channel: " << _fareRule.getChannel ());
00247 }
00248
00249 // //////////////////////////////////////
00250 storeAdvancePurchase ::
00251 storeAdvancePurchase (FareRuleStruct&
ioFareRule)
00252 : ParserSemanticAction (ioFareRule) {
00253 }
00254
00255 // //////////////////////////////////////
00256 void storeAdvancePurchase::operator() (
unsigned int iAdancePurchase,
00257                                       boost::spirit::qi::unused_type,
00258                                       boost::spirit::qi::unused_type)
const {
00259     const stdair::DayDuration_T& lAdancePurchase = iAdancePurchase;
00260     _fareRule.setAdvancePurchase (lAdancePurchase)
;
00261     // DEBUG
00262     //STDAIR_LOG_DEBUG ( "Advance Purchase: " << _fareRule.getAdvancePurchase
());
00263 }
00264
00265 // //////////////////////////////////////
00266 storeSaturdayStay ::
00267 storeSaturdayStay (FareRuleStruct&
ioFareRule)
00268 : ParserSemanticAction (ioFareRule) {
00269 }
00270
00271 // //////////////////////////////////////
00272 void storeSaturdayStay::operator() (char
iSaturdayStay,
00273                                     boost::spirit::qi::unused_type,
00274                                     boost::spirit::qi::unused_type) const {
00275     bool lBool = false;

```

```

00276         if (iSaturdayStay == 'T') {
00277             lBool = true;
00278         } else {
00279             if (iSaturdayStay != 'F') {
00280                 // DEBUG
00281                 STDAIR_LOG_DEBUG ("Invalid saturdayStay char " << iSaturdayStay);
00282             }
00283         }
00284         const stdair::SaturdayStay_T lSaturdayStay (lBool);
00285         _fareRule.setSaturdayStay (lSaturdayStay);
00286         // DEBUG
00287         //STDAIR_LOG_DEBUG ("Saturday Stay: " << _fareRule.getSaturdayStay ());
00288     }
00289
00290     // //////////////////////////////////////
00291     storeChangeFees ::
00292     storeChangeFees (FareRuleStruct&
ioFareRule)
00293     : ParserSemanticAction (ioFareRule) {
00294     }
00295
00296     // //////////////////////////////////////
00297     void storeChangeFees::operator() (char
iChangefees,
00298                                     boost::spirit::qi::unused_type,
00299                                     boost::spirit::qi::unused_type) const {
00300
00301         bool lBool = false;
00302         if (iChangefees == 'T') {
00303             lBool = true;
00304         } else {
00305             if (iChangefees != 'F') {
00306                 // DEBUG
00307                 STDAIR_LOG_DEBUG ("Invalid change fees char " << iChangefees);
00308             }
00309         }
00310         const stdair::ChangeFees_T lChangefees (lBool);
00311         _fareRule.setChangeFees (lChangefees);
00312         // DEBUG
00313         //STDAIR_LOG_DEBUG ("Change fees: " << _fareRule.getChangeFees ());
00314     }
00315
00316     // //////////////////////////////////////
00317     storeNonRefundable ::
00318     storeNonRefundable (FareRuleStruct&
ioFareRule)
00319     : ParserSemanticAction (ioFareRule) {
00320     }
00321
00322     // //////////////////////////////////////
00323     void storeNonRefundable::operator() (char
iNonRefundable,
00324                                     boost::spirit::qi::unused_type,
00325                                     boost::spirit::qi::unused_type) const
{
00326         bool lBool = false;
00327         if (iNonRefundable == 'T') {
00328             lBool = true;
00329         } else {
00330             if (iNonRefundable != 'F') {
00331                 // DEBUG
00332                 STDAIR_LOG_DEBUG ("Invalid non refundable char " << iNonRefundable);
00333             }
00334         }
00335         const stdair::NonRefundable_T lNonRefundable (lBool);
00336         _fareRule.setNonRefundable (lNonRefundable);
00337         // DEBUG
00338         //STDAIR_LOG_DEBUG ("Non refundable: " << _fareRule.getNonRefundable
());
00339     }
00340
00341     // //////////////////////////////////////
00342     storeMinimumStay ::
00343     storeMinimumStay (FareRuleStruct&
ioFareRule)
00344     : ParserSemanticAction (ioFareRule) {
00345     }
00346
00347     // //////////////////////////////////////
00348     void storeMinimumStay::operator() (unsigned
int iMinStay,
00349                                     boost::spirit::qi::unused_type,
00350                                     boost::spirit::qi::unused_type) const {
00351         const stdair::DayDuration_T lMinStay = iMinStay;
00352         _fareRule.setMinimumStay (lMinStay);
00353         // DEBUG
00354         //STDAIR_LOG_DEBUG ("Minimum Stay: " << _fareRule.getMinimumStay ());

```

```

00355     }
00356
00357     // //////////////////////////////////////
00358     storeFare ::
00359     storeFare (FareRuleStruct& ioFareRule)
00360         : ParserSemanticAction (ioFareRule) {
00361     }
00362
00363     // //////////////////////////////////////
00364     void storeFare::operator() (double iFare,
00365                                 boost::spirit::qi::unused_type,
00366                                 boost::spirit::qi::unused_type) const {
00367         const stdair::PriceValue_T lFare = iFare;
00368         _fareRule.setFare (lFare);
00369         // DEBUG
00370         //STDAIR_LOG_DEBUG ("Fare: " << _fareRule.getFare ());
00371     }
00372
00373     // //////////////////////////////////////
00374     storeAirlineCode ::
00375     storeAirlineCode (FareRuleStruct&
00376                       ioFareRule)
00377         : ParserSemanticAction (ioFareRule) {
00378     }
00379
00379     // //////////////////////////////////////
00380     void storeAirlineCode::operator() (
00381         std::vector<char> iChar,
00382                                     boost::spirit::qi::unused_type,
00383                                     boost::spirit::qi::unused_type) const {
00384         const stdair::AirlineCode_T lAirlineCode (iChar.begin(), iChar.end());
00385         // Insertion of this airline Code list in the whole AirlineCode name
00386         _fareRule.addAirlineCode (lAirlineCode);
00387         // DEBUG
00388         //STDAIR_LOG_DEBUG ( "Airline code: " << lAirlineCode);
00389     }
00390
00391     // //////////////////////////////////////
00392     storeClass ::
00393     storeClass (FareRuleStruct& ioFareRule)
00394         : ParserSemanticAction (ioFareRule) {
00395     }
00396
00397     // //////////////////////////////////////
00398     void storeClass::operator() (std::vector<char> iChar
00399
00399                                     boost::spirit::qi::unused_type,
00400                                     boost::spirit::qi::unused_type) const {
00401         std::ostringstream ostr;
00402         for (std::vector<char>::const_iterator lItVector = iChar.begin();
00403              lItVector != iChar.end();
00404              lItVector++) {
00405             ostr << *lItVector;
00406         }
00407         const std::string classCodeStr = ostr.str();
00408         const stdair::ClassCode_T lClassCode (classCodeStr);
00409         // Insertion of this class Code list in the whole classCode name
00410         _fareRule.addClassCode (lClassCode);
00411         // DEBUG
00412         //STDAIR_LOG_DEBUG ("Class Code: " << lClassCode);
00413     }
00414
00415     // //////////////////////////////////////
00416     doEndFare::
00417     doEndFare (stdair::BomRoot& ioBomRoot,
00418                FareRuleStruct& ioFareRule)
00419         : ParserSemanticAction (ioFareRule),
00420           _bomRoot (ioBomRoot) {
00421     }
00422
00423     // //////////////////////////////////////
00424     void doEndFare::operator() (
00425         boost::spirit::qi::unused_type,
00426         boost::spirit::qi::unused_type,
00427         boost::spirit::qi::unused_type) const {
00428         // DEBUG
00429         //STDAIR_LOG_DEBUG ("Do End");
00430         // Generation of the fare rule object.
00431         FareRuleGenerator::createAirportPair (_bomRoot, _fareRule
00432 );
00433         STDAIR_LOG_DEBUG(_fareRule.describe());
00434     }
00435
00436     // //////////////////////////////////////
00437     //
00438     // Utility Parsers

```

```

00437 //
00438 // //////////////////////////////////////
00440 namespace bsq = boost::spirit::qi;
00441 namespace bsa = boost::spirit::ascii;
00442
00444 stdair::int1_p_t int1_p;
00445
00447 stdair::uint2_p_t uint2_p;
00448
00450 stdair::uint4_p_t uint4_p;
00451
00453 stdair::uint1_4_p_t uint1_4_p;
00454
00456 stdair::hour_p_t hour_p;
00457 stdair::minute_p_t minute_p;
00458 stdair::second_p_t second_p;
00459
00461 stdair::year_p_t year_p;
00462 stdair::month_p_t month_p;
00463 stdair::day_p_t day_p;
00464
00466 //
00467 // (Boost Spirit) Grammar Definition
00468 //
00470
00499 template <typename Iterator>
00500 struct FareRuleParser :
00501     public boost::spirit::qi::grammar<Iterator,
00502         boost::spirit::ascii::space_type> {
00503
00504     FareRuleParser (stdair::BomRoot& ioBomRoot,
00505         FareRuleStruct& iofareRule) :
00506
00507         FareRuleParser::base_type(start),
00508         _bomRoot(ioBomRoot), _fareRule(iofareRule) {
00509
00510
00511         start = *(comments | fare_rule);
00512
00513         comments = (bsq::lexeme[bsq::repeat(2)[bsa::char_('/')]]
00514             >> +(bsa::char_ - bsq::eol)
00515             >> bsq::eol]
00516             | bsq::lexeme[bsa::char_('/') >> bsa::char_('*')
00517             >> +(bsa::char_ - bsa::char_('*'))
00518             >> bsa::char_('*') >> bsa::char_('/')]);
00519
00520         fare_rule = fare_key
00521             >> +( ';' >> segment )
00522             >> fare_rule_end[doEndFare(_bomRoot,
00523                 _fareRule)];
00524
00525         fare_rule_end = bsa::char_(';');
00526
00527         fare_key = fare_id
00528             >> ';' >> origin >> ';' >> destination
00529             >> ';' >> tripType
00530             >> ';' >> dateRangeStart >> ';' >> dateRangeEnd
00531             >> ';' >> timeRangeStart >> ';' >> timeRangeEnd
00532             >> ';' >> point_of_sale >> ';' >> cabinCode >>
00533             ';' >> channel
00534             >> ';' >> advancePurchase >> ';' >> saturdayStay
00535             >> ';' >> changeFees >> ';' >> nonRefundable
00536             >> ';' >> minimumStay >> ';' >> fare;
00537
00538         fare_id = uint1_4_p[storeFareId(_fareRule
00539             )];
00540
00541         origin = bsq::repeat(3)[bsa::char_("A-Z")][storeOrigin(
00542             _fareRule)];
00543
00544         destination =
00545             bsq::repeat(3)[bsa::char_("A-Z")][storeDestination(
00546                 _fareRule)];
00547
00548         tripType =
00549             bsq::repeat(2)[bsa::char_("A-Z")][storeTripType(_fareRule
00550                 )];
00551
00552         dateRangeStart = date[storeDateRangeStart(
00553             _fareRule)];
00554
00555         dateRangeEnd = date[storeDateRangeEnd(
00556             _fareRule)];
00557
00558         date = bsq::lexeme
00559             [year_p[boost::phoenix::ref(_fareRule._itYear) =
00560                 bsq::labels::_1]

```



```

00552         >> '-'
00553         >> month_p[boost::phoenix::ref(_fareRule._itMonth
) = bsq::labels::_1]
00554         >> '-'
00555         >> day_p[boost::phoenix::ref(_fareRule._itDay) =
bsq::labels::_1] ];
00556
00557         timeRangeStart = time[storeStartTime
(_fareRule)];
00558
00559         timeRangeEnd = time[storeEndTime(
_fareRule)];
00560
00561         time = bsq::lexeme
00562         [hour_p[boost::phoenix::ref(_fareRule._itHours)
= bsq::labels::_1]
00563         >> ':'
00564         >> minute_p[boost::phoenix::ref(_fareRule._itMinutes
) = bsq::labels::_1]
00565         >> - (':' >> second_p[boost::phoenix::ref(_fareRule.
_itSeconds) = bsq::labels::_1]) ];
00566
00567         point_of_sale = bsq::repeat(3) [bsa::char_("A-Z")] [storePOS
(_fareRule)];
00568
00569         cabinCode = bsa::char_("A-Z") [storeCabinCode(
_fareRule)];
00570
00571         channel = bsq::repeat(2) [bsa::char_("A-Z")] [storeChannel
(_fareRule)];
00572
00573         advancePurchase = uint1_4_p[storeAdvancePurchase
(_fareRule)];
00574
00575         saturdayStay = bsa::char_("A-Z") [storeSaturdayStay
(_fareRule)];
00576
00577         changeFees = bsa::char_("A-Z") [storeChangeFees (
_fareRule)];
00578
00579         nonRefundable = bsa::char_("A-Z") [storeNonRefundable
(_fareRule)];
00580
00581         minimumStay = uint1_4_p[storeMinimumStay
(_fareRule)];
00582
00583         fare = bsq::double_[storeFare(_fareRule)];
00584
00585         segment = bsq::repeat(2) [bsa::char_("A-Z")] [storeAirlineCode
(_fareRule)]
00586         >> ';'
00587         >> bsq::repeat(1,bsq::inf) [bsa::char_("A-Z")] [storeClass (
_fareRule)];
00588
00589         //BOOST_SPIRIT_DEBUG_NODE (FareRuleParser);
00590         BOOST_SPIRIT_DEBUG_NODE (start);
00591         BOOST_SPIRIT_DEBUG_NODE (comments);
00592         BOOST_SPIRIT_DEBUG_NODE (fare_rule);
00593         BOOST_SPIRIT_DEBUG_NODE (fare_rule_end);
00594         BOOST_SPIRIT_DEBUG_NODE (fare_key);
00595         BOOST_SPIRIT_DEBUG_NODE (fare_id);
00596         BOOST_SPIRIT_DEBUG_NODE (origin);
00597         BOOST_SPIRIT_DEBUG_NODE (destination);
00598         BOOST_SPIRIT_DEBUG_NODE (tripType);
00599         BOOST_SPIRIT_DEBUG_NODE (dateRangeStart);
00600         BOOST_SPIRIT_DEBUG_NODE (dateRangeEnd);
00601         BOOST_SPIRIT_DEBUG_NODE (date);
00602         BOOST_SPIRIT_DEBUG_NODE (timeRangeStart);
00603         BOOST_SPIRIT_DEBUG_NODE (time);
00604         BOOST_SPIRIT_DEBUG_NODE (point_of_sale);
00605         BOOST_SPIRIT_DEBUG_NODE (cabinCode);
00606         BOOST_SPIRIT_DEBUG_NODE (channel);
00607         BOOST_SPIRIT_DEBUG_NODE (advancePurchase);
00608         BOOST_SPIRIT_DEBUG_NODE (saturdayStay);
00609         BOOST_SPIRIT_DEBUG_NODE (changeFees);
00610         BOOST_SPIRIT_DEBUG_NODE (nonRefundable);
00611         BOOST_SPIRIT_DEBUG_NODE (minimumStay);
00612         BOOST_SPIRIT_DEBUG_NODE (fare);
00613         BOOST_SPIRIT_DEBUG_NODE (segment);
00614
00615     }
00616
00617     // Instantiation of rules
00618     boost::spirit::qi::rule<Iterator,
00619         boost::spirit::ascii::space_type>
00620     start, comments, fare_rule, fare_rule_end
, fare_key, fare_id, origin,

```

```

00621     destination, tripType, dateRangeStart,
00622     dateRangeEnd, date,
00623     timeRangeStart, timeRangeEnd, time,
00624     point_of_sale, cabinCode, channel,
00625     advancePurchase, saturdayStay, changeFees
00626     , nonRefundable, minimumStay,
00627     fare, segment;
00628
00629     // Parser Context
00630     stdair::BomRoot& _bomRoot;
00631     FareRuleStruct& _fareRule;
00632 };
00633
00634 // Entry class for the file parser
00635 //
00636 //
00637 //
00638 //
00639 //
00640 //
00641 FareRuleFileParser::
00642 FareRuleFileParser (stdair::BomRoot& ioBomRoot,
00643                     const stdair::Filename_T& iFilename)
00644 : _filename (iFilename), _bomRoot (ioBomRoot) {
00645     init();
00646 }
00647
00648 //
00649 void FareRuleFileParser::init() {
00650     // Check that the file exists and is readable
00651     const bool doesExistAndIsReadable =
00652         stdair::BasFileMgr::doesExistAndIsReadable (_filename);
00653
00654     if (doesExistAndIsReadable == false) {
00655         STDAIR_LOG_ERROR ("The fare schedule file " << _filename
00656                         << " does not exist or can not be read.");
00657
00658         throw FareInputFileNotFoundException ("The
00659 fare file " + _filename
00660                                         + " does not exist or can not be
00661 read");
00662     }
00663
00664 //
00665 void FareRuleFileParser::generateFareRules
00666 () {
00667     STDAIR_LOG_DEBUG ("Parsing fare input file: " << _filename);
00668
00669     // File to be parsed
00670     const std::string* lFileName = &_filename;
00671     const char *lChar = (*lFileName).c_str();
00672     std::ifstream fileToBeParsed(lChar, std::ios_base::in);
00673
00674     // Check if the filename exist and can be open
00675     if (fileToBeParsed == false) {
00676         STDAIR_LOG_ERROR ("The fare file " << _filename << " can not be open."
00677                         << std::endl);
00678
00679         throw FareInputFileNotFoundException ("The
00680 file " + _filename
00681                                         + " does not exist or can not be
00682 read");
00683     }
00684
00685     // Create an input iterator
00686     stdair::base_iterator_t inputBegin (fileToBeParsed);
00687
00688     // Convert input iterator to an iterator usable by spirit parser
00689     stdair::iterator_t
00690         start (boost::spirit::make_default_multi_pass (inputBegin));
00691     stdair::iterator_t end;
00692
00693     // Initialise the parser (grammar) with the helper/staging structure.
00694     FareParserHelper::FareRuleParser<stdair::iterator_t>
00695     lFPParser(_bomRoot, _fareRule);
00696
00697     // Launch the parsing of the file and, thanks to the doEndFare
00698     // call-back structure, the building of the whole BomRoot BOM
00699     const bool hasParsingBeenSuccessful =
00700         boost::spirit::qi::phrase_parse (start, end, lFPParser,
00701                                         boost::spirit::ascii::space);
00702
00703     if (hasParsingBeenSuccessful == false) {
00704         STDAIR_LOG_ERROR ("Parsing of fare input file: " << _filename

```

```

00701             << " failed");
00702         throw FareFileParsingFailedException ("
Parsing of fare input file: "
00703             + _filename + " failed");
00704     }
00705
00706     if (start != end) {
00707         STDAIR_LOG_ERROR ("Parsing of fare input file: " << _filename
00708             << " failed");
00709         throw FareFileParsingFailedException ("
Parsing of fare input file: "
00710             + _filename + " failed");
00711     }
00712
00713     if (hasParsingBeenSuccessful == true && start == end) {
00714         STDAIR_LOG_DEBUG ("Parsing of fare input file: " << _filename
00715             << " succeeded");
00716     }
00717
00718 }
00719
00720 }

```

23.33 simfqt/command/FareParserHelper.hpp File Reference

```

#include <string>
#include <boost/spirit/include/qi.hpp>
#include <stdair/command/CmdAbstract.hpp>
#include <simfqt/SIMFQT_Types.hpp>
#include <simfqt/bom/FareRuleStruct.hpp>

```

Classes

- struct [SIMFQT::FareParserHelper::ParserSemanticAction](#)
- struct [SIMFQT::FareParserHelper::storeFareId](#)
- struct [SIMFQT::FareParserHelper::storeOrigin](#)
- struct [SIMFQT::FareParserHelper::storeDestination](#)
- struct [SIMFQT::FareParserHelper::storeTripType](#)
- struct [SIMFQT::FareParserHelper::storeDateRangeStart](#)
- struct [SIMFQT::FareParserHelper::storeDateRangeEnd](#)
- struct [SIMFQT::FareParserHelper::storeStartRangeTime](#)
- struct [SIMFQT::FareParserHelper::storeEndRangeTime](#)
- struct [SIMFQT::FareParserHelper::storePOS](#)
- struct [SIMFQT::FareParserHelper::storeCabinCode](#)
- struct [SIMFQT::FareParserHelper::storeChannel](#)
- struct [SIMFQT::FareParserHelper::storeAdvancePurchase](#)
- struct [SIMFQT::FareParserHelper::storeSaturdayStay](#)
- struct [SIMFQT::FareParserHelper::storeChangeFees](#)
- struct [SIMFQT::FareParserHelper::storeNonRefundable](#)
- struct [SIMFQT::FareParserHelper::storeMinimumStay](#)
- struct [SIMFQT::FareParserHelper::storeFare](#)
- struct [SIMFQT::FareParserHelper::storeAirlineCode](#)
- struct [SIMFQT::FareParserHelper::storeClass](#)
- struct [SIMFQT::FareParserHelper::doEndFare](#)
- class [SIMFQT::FareRuleFileParser](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [SIMFQT](#)
- namespace [SIMFQT::FareParserHelper](#)

23.34 FareParserHelper.hpp

```

00001 #ifndef __SIMFQT_CMD_FAREPARSERHELPER_HPP
00002 #define __SIMFQT_CMD_FAREPARSERHELPER_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 // Simfqt
00014 #include <simfqt/SIMFQT_Types.hpp>
00015 #include <simfqt/bom/FareRuleStruct.hpp>
00016
00017 // Forward declarations
00018 namespace stdair {
00019     class BomRoot;
00020 }
00021
00022 namespace SIMFQT {
00023
00024     namespace FareParserHelper {
00025
00026         // //////////////////////////////////////
00027         // Semantic actions
00028         // //////////////////////////////////////
00029
00030         struct ParserSemanticAction {
00031             ParserSemanticAction (FareRuleStruct&);
00032             FareRuleStruct& _fareRule;
00033         };
00034
00035         struct storeFareId : public ParserSemanticAction
00036         {
00037             storeFareId (FareRuleStruct&);
00038             void operator() (unsigned int,
00039                             boost::spirit::qi::unused_type,
00040                             boost::spirit::qi::unused_type) const;
00041         };
00042
00043         struct storeOrigin : public ParserSemanticAction
00044         {
00045             storeOrigin (FareRuleStruct&);
00046             void operator() (std::vector<char>,
00047                             boost::spirit::qi::unused_type,
00048                             boost::spirit::qi::unused_type) const;
00049         };
00050
00051         struct storeDestination : public ParserSemanticAction
00052         {
00053             storeDestination (FareRuleStruct&);
00054             void operator() (std::vector<char>,
00055                             boost::spirit::qi::unused_type,
00056                             boost::spirit::qi::unused_type) const;
00057         };
00058
00059         struct storeTripType : public ParserSemanticAction
00060         {
00061             storeTripType (FareRuleStruct&);
00062             void operator() (std::vector<char>,
00063                             boost::spirit::qi::unused_type,
00064                             boost::spirit::qi::unused_type) const;
00065         };
00066
00067         struct storeDateRangeStart : public ParserSemanticAction
00068         {
00069             storeDateRangeStart (FareRuleStruct&);
00070             void operator() (boost::spirit::qi::unused_type,
00071                             boost::spirit::qi::unused_type,
00072                             boost::spirit::qi::unused_type) const;
00073         };
00074
00075         struct storeDateRangeEnd : public ParserSemanticAction
00076         {
00077             storeDateRangeEnd (FareRuleStruct&);
00078             void operator() (boost::spirit::qi::unused_type,
00079                             boost::spirit::qi::unused_type,
00080                             boost::spirit::qi::unused_type) const;
00081         };
00082
00083         struct storeStartRangeTime : public ParserSemanticAction
00084         {

```

```

00102     storeStartRangeTime (FareRuleStruct&);
00104     void operator() (boost::spirit::qi::unused_type,
00105                     boost::spirit::qi::unused_type,
00106                     boost::spirit::qi::unused_type) const;
00107 };
00108
00110 struct storeEndRangeTime : public ParserSemanticAction
00111 {
00112     storeEndRangeTime (FareRuleStruct&);
00114     void operator() (boost::spirit::qi::unused_type,
00115                     boost::spirit::qi::unused_type,
00116                     boost::spirit::qi::unused_type) const;
00117 };
00118
00120 struct storePOS : public ParserSemanticAction {
00122     storePOS (FareRuleStruct&);
00124     void operator() (std::vector<char>,
00125                     boost::spirit::qi::unused_type,
00126                     boost::spirit::qi::unused_type) const;
00127 };
00128
00130 struct storeCabinCode : public ParserSemanticAction
00131 {
00132     storeCabinCode (FareRuleStruct&);
00134     void operator() (char,
00135                     boost::spirit::qi::unused_type,
00136                     boost::spirit::qi::unused_type) const;
00137 };
00138
00140 struct storeChannel : public ParserSemanticAction
00141 {
00142     storeChannel (FareRuleStruct&);
00144     void operator() (std::vector<char>,
00145                     boost::spirit::qi::unused_type,
00146                     boost::spirit::qi::unused_type) const;
00147 };
00148
00150 struct storeAdvancePurchase : public
00151 ParserSemanticAction {
00152     storeAdvancePurchase (FareRuleStruct&);
00154     void operator() (unsigned int,
00155                     boost::spirit::qi::unused_type,
00156                     boost::spirit::qi::unused_type) const;
00157 };
00158
00160 struct storeSaturdayStay : public ParserSemanticAction
00161 {
00162     storeSaturdayStay (FareRuleStruct&);
00164     void operator() (char,
00165                     boost::spirit::qi::unused_type,
00166                     boost::spirit::qi::unused_type) const;
00167 };
00168
00170 struct storeChangeFees : public ParserSemanticAction
00171 {
00172     storeChangeFees (FareRuleStruct&);
00174     void operator() (char,
00175                     boost::spirit::qi::unused_type,
00176                     boost::spirit::qi::unused_type) const;
00177 };
00178
00180 struct storeNonRefundable : public ParserSemanticAction
00181 {
00182     storeNonRefundable (FareRuleStruct&);
00184     void operator() (char,
00185                     boost::spirit::qi::unused_type,
00186                     boost::spirit::qi::unused_type) const;
00187 };
00188
00190 struct storeMinimumStay : public ParserSemanticAction
00191 {
00192     storeMinimumStay (FareRuleStruct&);
00194     void operator() (unsigned int,
00195                     boost::spirit::qi::unused_type,
00196                     boost::spirit::qi::unused_type) const;
00197 };
00198
00200 struct storeFare : public ParserSemanticAction
00201 {
00202     storeFare (FareRuleStruct&);
00204     void operator() (double,
00205                     boost::spirit::qi::unused_type,
00206                     boost::spirit::qi::unused_type) const;
00207 };
00208
00210 struct storeAirlineCode : public ParserSemanticAction
00211 {

```

```

00212     storeAirlineCode (FareRuleStruct&);
00214     void operator() (std::vector<char>,
00215                     boost::spirit::qi::unused_type,
00216                     boost::spirit::qi::unused_type) const;
00217 };
00218
00220 struct storeClass : public ParserSemanticAction
00221 {
00222     storeClass (FareRuleStruct&);
00224     void operator() (std::vector<char>,
00225                     boost::spirit::qi::unused_type,
00226                     boost::spirit::qi::unused_type) const;
00227 };
00228
00230 struct doEndFare : public ParserSemanticAction
00231 {
00232     doEndFare (stdair::BomRoot&, FareRuleStruct&);
00234     void operator() (boost::spirit::qi::unused_type,
00235                     boost::spirit::qi::unused_type,
00236                     boost::spirit::qi::unused_type) const;
00238     stdair::BomRoot& _bomRoot;
00239 };
00240
00241 }
00242
00244 //
00245 // Entry class for the file parser
00246 //
00248
00254 class FareRuleFileParser : public stdair::CmdAbstract {
00255 public:
00257     FareRuleFileParser (stdair::BomRoot& ioBomRoot,
00258                         const stdair::Filename_T& iFilename);
00259
00261     void generateFareRules ();
00262
00263 private:
00265     void init();
00266
00267 private:
00268     // Attributes
00270     stdair::Filename_T _filename;
00271
00273     stdair::BomRoot& _bomRoot;
00274
00276     FareRuleStruct _fareRule;
00277 };
00278
00279 }
00280 #endif // __SIMFQT_CMD_FAREPARSERHELPER_HPP

```

23.35 simfq/command/FareQuoter.cpp File Reference

```
#include <cassert>
```

```

#include <sstream>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/ParsedKey.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/FlightDateKey.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/FareFeatures.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/key_types.hpp>
#include <simfqt/SIMFQT_Types.hpp>
#include <simfqt/command/FareQuoter.hpp>

```

Namespaces

- namespace [SIMFQT](#)

23.36 FareQuoter.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BomDisplay.hpp>
00009 #include <stdair/bom/BomKeyManager.hpp>
00010 #include <stdair/bom/ParsedKey.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/BomRoot.hpp>
00013 #include <stdair/bom/InventoryKey.hpp>
00014 #include <stdair/bom/FlightDateKey.hpp>
00015 #include <stdair/bom/SegmentDateKey.hpp>
00016 #include <stdair/bom/AirlineClassList.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/FareFeatures.hpp>
00022 #include <stdair/bom/BookingRequestStruct.hpp>
00023 #include <stdair/bom/TravelSolutionStruct.hpp>
00024 #include <stdair/service/Logger.hpp>
00025 #include <stdair/bom/key_types.hpp>
00026 // SimFQT
00027 #include <simfqt/SIMFQT_Types.hpp>
00028 #include <simfqt/command/FareQuoter.hpp>
00029
00030 namespace SIMFQT {
00031
00032     bool FareQuoter::_atLeastOneAvailableDateRule = false;
00033     bool FareQuoter::_atLeastOneAvailablePosChannel = false;
00034     bool FareQuoter::_atLeastOneAvailableTimeRule = false;
00035     bool FareQuoter::_atLeastOneAvailableFeaturesRule = false;
00036     bool FareQuoter::_atLeastOneAvailableAirlineClassRule = false;
00037
00038     // //////////////////////////////////////
00039     FareQuoter::FareQuoter() {
00040         assert (false);
00041     }
00042

```

```

00043 ///////////////////////////////////////////////////////////////////
00044 FareQuoter::FareQuoter(const FareQuoter&) {
00045     assert (false);
00046 }
00047
00048 ///////////////////////////////////////////////////////////////////
00049 FareQuoter::~FareQuoter() {
00050 }
00051
00052 ///////////////////////////////////////////////////////////////////
00053 void FareQuoter::reset() {
00054     _atLeastOneAvailableDateRule = false;
00055     _atLeastOneAvailablePosChannel = false;
00056     _atLeastOneAvailableTimeRule = false;
00057     _atLeastOneAvailableFeaturesRule = false;
00058     _atLeastOneAvailableAirlineClassRule = false;
00059 }
00060
00061 ///////////////////////////////////////////////////////////////////
00062 void FareQuoter::
00063 priceQuote (const stdair::BookingRequestStruct& iBookingRequest,
00064             stdair::TravelSolutionList_T& ioTravelSolutionList,
00065             const stdair::BomRoot& iBomRoot) {
00066
00067     // Do an independent price quote for each travel solution related to the
00068     // booking request.
00069     for (stdair::TravelSolutionList_T::iterator itTravelSolution =
00070          ioTravelSolutionList.begin();
00071          itTravelSolution != ioTravelSolutionList.end(); ++itTravelSolution) {
00072         reset();
00073         // Select a travel solution.
00074         stdair::TravelSolutionStruct& lTravelSolutionStruct = *itTravelSolution;
00075         // Price quote the travel solution into question.
00076         priceQuote (iBookingRequest, lTravelSolutionStruct, iBomRoot);
00077     }
00078 }
00079
00080 ///////////////////////////////////////////////////////////////////
00081 void FareQuoter::
00082 priceQuote (const stdair::BookingRequestStruct& iBookingRequest,
00083             stdair::TravelSolutionStruct& ioTravelSolution,
00084             const stdair::BomRoot& iBomRoot) {
00085
00086     // Get the origin of the first segment in order to get the origin of
00087     // the solution.
00088     const stdair::ParsedKey& lFirstSegmentKey =
00089         getFirstSPParsedKey(ioTravelSolution);
00090     const stdair::AirportCode_T& lOrigin = lFirstSegmentKey._boardingPoint;
00091
00092     // Get the destination of the last segment in order to get the
00093     // destination of the solution.
00094     const stdair::ParsedKey& lLastSegmentKey =
00095         getLastSPParsedKey(ioTravelSolution);
00096     const stdair::AirportCode_T& lDestination = lLastSegmentKey._offPoint;
00097
00098     // Construct the Airport pair stream of the segment path.
00099     const stdair::AirportPairKey lAirportPairKey (lOrigin, lDestination);
00100
00101     // Search for the fare rules having the same origin and destination
00102     airports
00103     // as the travel solution
00104     const stdair::AirportPair* lAirportPair_ptr = stdair::BomManager::
00105         getObjectPtr<stdair::AirportPair> (iBomRoot, lAirportPairKey.toString());
00106
00107     // If no fare rule has the same origin and destination airports, the
00108     // pricing
00109     // is not possible, throw an exception.
00110     if (lAirportPair_ptr == NULL) {
00111         STDAIR_LOG_ERROR ("No available fare rule for the "
00112                          "<< "Origin-Destination pair: "
00113                          "<< lAirportPairKey.toString());
00114         throw AirportPairNotFoundException ("No available fare rule for "
00115                                             "the Origin-Destination pair: "
00116                                             + lAirportPairKey.toString());
00117     }
00118     // Sanity check.
00119     assert(lAirportPair_ptr != NULL);
00120
00121     // Fare rule(s) with the same origin and destination airports exist(s), now
00122     // the date range need to be checked.
00123     const stdair::AirportPair& lAirportPair = *lAirportPair_ptr;
00124     priceQuote(iBookingRequest, ioTravelSolution, lAirportPair);
00125
00126     if (_atLeastOneAvailableAirlineClassRule == false) {
00127         displayMissingFareRuleMessage (iBookingRequest, ioTravelSolution);
00128     }

```



```

00127     }
00128 }
00129
00130 // //////////////////////////////////////
00131 void FareQuoter::
00132 priceQuote (const stdair::BookingRequestStruct& iBookingRequest,
00133             stdair::TravelSolutionStruct& ioTravelSolution,
00134             const stdair::AirportPair& iAirportPair) {
00135
00136     // Get the first segment path parsed key.
00137     const stdair::ParsedKey lFirstSPParsedKey =
00138         getFirstSPParsedKey(ioTravelSolution);
00139
00140     // Get the date of the first segment date key.
00141     const stdair::FlightDateKey& lFlightDateKey =
00142         lFirstSPParsedKey.getFlightDateKey();
00143     const stdair::Date_T& lSPDate = lFlightDateKey.getDepartureDate();
00144
00145     // Get the list of the fare date ranges.
00146     const stdair::DatePeriodList_T& lFareDatePeriodList =
00147         stdair::BomManager::getList<stdair::DatePeriod> (iAirportPair);
00148
00149     // Browse the list of the fare rules date range.
00150     for (stdair::DatePeriodList_T::const_iterator itDateRange =
00151         lFareDatePeriodList.begin();
00152         itDateRange != lFareDatePeriodList.end(); ++itDateRange) {
00153
00154         const stdair::DatePeriod* lCurrentFareDatePeriod_ptr = *itDateRange ;
00155         assert (lCurrentFareDatePeriod_ptr != NULL);
00156
00157         // Select the fare rules having a corresponding date range.
00158         const bool isDepartureDateValid =
00159             lCurrentFareDatePeriod_ptr->isDepartureDateValid (lSPDate);
00160
00161         // If a fare rule has a corresponding date range, its channel and
00162         position
00163         // need to be checked.
00164         if (isDepartureDateValid == true) {
00165             _atLeastOneAvailableDateRule = true;
00166             const stdair::DatePeriod& lCurrentFareDatePeriod =
00167                 *lCurrentFareDatePeriod_ptr;
00168             priceQuote (iBookingRequest, ioTravelSolution,
00169                         lCurrentFareDatePeriod, iAirportPair);
00169         }
00170     }
00171 }
00172
00173 // //////////////////////////////////////
00174 void FareQuoter::
00175 priceQuote (const stdair::BookingRequestStruct& iBookingRequest,
00176             stdair::TravelSolutionStruct& ioTravelSolution,
00177             const stdair::DatePeriod& iFareDatePeriod,
00178             const stdair::AirportPair& iAirportPair) {
00179
00180     // Get the point-of-sale of the booking request.
00181     const stdair::CityCode_T& lPointOfSale = iBookingRequest.getPOS();
00182
00183     // Get the booking request channel.
00184     const stdair::ChannelLabel_T& lChannel =
00185         iBookingRequest.getBookingChannel();
00186
00187     // Construct the corresponding POS-channel primary key.
00188     const stdair::PosChannelKey lFarePosChannelKey (lPointOfSale, lChannel);
00189
00190     // Search for the fare rules having the same point-of-sale and channel as
00191     // the travel solution.
00192     const stdair::PosChannelList_T lFarePosChannelList =
00193         stdair::BomManager::getList<stdair::PosChannel> (iFareDatePeriod);
00194
00195     // Browse the list of the fare rules pos channel.
00196     for (stdair::PosChannelList_T::const_iterator itPosChannel =
00197         lFarePosChannelList.begin();
00198         itPosChannel != lFarePosChannelList.end();
00199         ++itPosChannel) {
00200         const stdair::PosChannel* lCurrentFarePosChannel_ptr = *itPosChannel;
00201         assert (lCurrentFarePosChannel_ptr != NULL);
00202
00203         // Get the point-of-sale and channel of the current fare rule.
00204         const stdair::CityCode_T& lCurrentPointOfSale =
00205             lCurrentFarePosChannel_ptr->getPos();
00206         const stdair::ChannelLabel_T& lCurrentChannel =
00207             lCurrentFarePosChannel_ptr->getChannel();
00208
00209         // Select the fare rules having a corresponding pos channel.
00210         if (lCurrentPointOfSale == lPointOfSale &&
00211             lCurrentChannel == lChannel) {
00212

```

```

00213         _atLeastOneAvailablePosChannel = true;
00214         // Fare rule(s) with the same point-of-sale and channel exist(s), now
00215         // the time range need to be checked.
00216         const stdair::PosChannel& lFarePosChannel= *lCurrentFarePosChannel_ptr;
00217         priceQuote (iBookingRequest, ioTravelSolution, lFarePosChannel);
00218     }
00219 }
00220
00221 }
00222
00223 // //////////////////////////////////////
00224 void FareQuoter::
00225 priceQuote (const stdair::BookingRequestStruct& iBookingRequest,
00226            stdair::TravelSolutionStruct& ioTravelSolution,
00227            const stdair::PosChannel& iFarePosChannel) {
00228
00229     // Get the first segment path parsed key.
00230     const stdair::ParsedKey lFirstSPParsedKey =
00231         getFirstSPParsedKey(ioTravelSolution);
00232
00233     // Get the segment boarding time of the segment path.
00234     const stdair::Duration_T& lSPTime = lFirstSPParsedKey.getBoardingTime();
00235
00236     // Get the list of the fare rules time period.
00237     const stdair::TimePeriodList_T& lFareTimePeriodList =
00238         stdair::BomManager::getList<stdair::TimePeriod> (iFarePosChannel);
00239
00240     // Browse the list of the fare rules time range.
00241     for (stdair::TimePeriodList_T::const_iterator itTimeRange =
00242          lFareTimePeriodList.begin();
00243          itTimeRange != lFareTimePeriodList.end();
00244          ++itTimeRange) {
00245         const stdair::TimePeriod* lCurrentFareTimePeriod_ptr = *itTimeRange ;
00246         assert (lCurrentFareTimePeriod_ptr != NULL);
00247
00248         // Select the fare rules having a corresponding time range.
00249         const bool isDepartureTimeValid =
00250             lCurrentFareTimePeriod_ptr->isDepartureTimeValid (lSPTime);
00251
00252         // If a fare rule has a corresponding time range, its advanced purchase,
00253         // trip type and minimum stay duration need to be checked.
00254         if (isDepartureTimeValid) {
00255             _atLeastOneAvailableTimeRule = true;
00256             const stdair::TimePeriod& lCurrentFareTimePeriod =
00257                 *lCurrentFareTimePeriod_ptr;
00258             priceQuote (iBookingRequest, ioTravelSolution,
00259                         lCurrentFareTimePeriod, iFarePosChannel);
00260         }
00261     }
00262 }
00263
00264 // //////////////////////////////////////
00265 void FareQuoter::
00266 priceQuote (const stdair::BookingRequestStruct& iBookingRequest,
00267            stdair::TravelSolutionStruct& ioTravelSolution,
00268            const stdair::TimePeriod& iFareTimePeriod,
00269            const stdair::PosChannel& iFarePosChannel) {
00270
00271     // Get the stay duration of the booking request.
00272     const stdair::DayDuration_T& lStayDuration=
00273         iBookingRequest.getStayDuration();
00274
00275     // Get the booking request trip type.
00276     const stdair::TripType_T& lTripType =
00277         iBookingRequest.getTripType();
00278
00279     // Get the booking request date time.
00280     const stdair::DateTime_T& lRequestDateTime =
00281         iBookingRequest.getRequestDateTime();
00282
00283     // Get the referenced departure date of the segment path.
00284     const stdair::ParsedKey lFirstSPParsedKey =
00285         getFirstSPParsedKey(ioTravelSolution);
00286     const stdair::Date_T& lSPDate =
00287         lFirstSPParsedKey.getFlightDateKey().getDepartureDate();
00288
00289     // Get the segment boarding time of the segment path.
00290     const stdair::Duration_T& lSPTime = lFirstSPParsedKey.getBoardingTime();
00291
00292     // Construct the date-time type correponding to the flight date
00293     const stdair::DateTime_T lSPDateTime (lSPDate, lSPTime);
00294
00295     bool isTripTypeValid = false;
00296     bool isStayDurationValid = false;
00297     bool isAdvancePurchaseValid = false;
00298
00299

```

```

00300 // Get the list of the fare features.
00301 const stdair::FareFeaturesList_T& lFareFeaturesList =
00302     stdair::BomManager::getList<stdair::FareFeatures> (iFareTimePeriod);
00303
00304 // Browse the list of the fare rules features.
00305 for (stdair::FareFeaturesList_T::const_iterator itFareFeatures =
00306     lFareFeaturesList.begin();
00307     itFareFeatures != lFareFeaturesList.end();
00308     ++itFareFeatures) {
00309     const stdair::FareFeatures* lCurrentFareFeatures_ptr =
00310         *itFareFeatures;
00311     assert (lCurrentFareFeatures_ptr != NULL);
00312
00313     // Does the current fare features correspond to a correct trip
00314     // type?
00315     isTripTypeValid =
00316         lCurrentFareFeatures_ptr->isTripTypeValid (lTripType);
00317     // Does the current fare features correspond to a correct stay
00318     // duration?
00319     isStayDurationValid =
00320         lCurrentFareFeatures_ptr->isStayDurationValid (lStayDuration);
00321     // Does the current fare features correspond to a correct advanced
00322     // purchase?
00323     isAdvancePurchaseValid = lCurrentFareFeatures_ptr->
00324         isAdvancePurchaseValid (lRequestDateTime,
00325             lSPDateTime);
00326
00327     // Search for the fare rules having corresponding features.
00328     if (isStayDurationValid && isAdvancePurchaseValid && isTripTypeValid){
00329         _atLeastOneAvailableFeaturesRule = true;
00330         // Create a fare structure for the travel solution.
00331         stdair::FareOptionStruct lFareOption;
00332         const stdair::ChangeFees_T& lChangeFees =
00333             lCurrentFareFeatures_ptr->getChangeFees();
00334         // Set the fare change fees.
00335         lFareOption.setChangeFees (lChangeFees);
00336         const stdair::NonRefundable_T& lNonRefundable =
00337             lCurrentFareFeatures_ptr->getRefundableOption();
00338         // Set the fare refundable option.
00339         lFareOption.setNonRefundable (lNonRefundable);
00340         const stdair::SaturdayStay_T& lSaturdayStay =
00341             lCurrentFareFeatures_ptr->getSaturdayStay();
00342         // Set the fare saturday night stay option.
00343         lFareOption.setSaturdayStay (lSaturdayStay);
00344         const stdair::FareFeatures& lCurrentFareFeatures =
00345             *lCurrentFareFeatures_ptr;
00346         priceQuote (iBookingRequest, ioTravelSolution,
00347             lCurrentFareFeatures, iFarePosChannel,
00348             lFareOption);
00349     }
00350 }
00351
00352 }
00353
00354
00355 // //////////////////////////////////////
00356 void FareQuoter::
00357 priceQuote (const stdair::BookingRequestStruct& iBookingRequest,
00358     stdair::TravelSolutionStruct& ioTravelSolution,
00359     const stdair::FareFeatures& iFareFeatures,
00360     const stdair::PosChannel& iFarePosChannel,
00361     stdair::FareOptionStruct& iFareOption) {
00362
00363     // Get the first segment path parsed key.
00364     const stdair::ParsedKey lFirstSPParsedKey =
00365         getFirstSPParsedKey (ioTravelSolution);
00366
00367     // Get the segment-path of the travel solution.
00368     const stdair::SegmentPath_T& lSegmentPath =
00369         ioTravelSolution.getSegmentPath();
00370
00371     // Get the list of the fare rules.
00372     const stdair::AirlineClassListList_T& lAirlineClassListList =
00373         stdair::BomManager::getList<stdair::AirlineClassList> (iFareFeatures);
00374
00375     bool lCorrectAirlineRule = false;
00376     bool lAtLeastOneDifferentAirline = false;
00377
00378     // Browse the list of airline code list and search for the fare rules
00379     // having a corresponding airline list.
00380     for (stdair::AirlineClassListList_T::const_iterator itAirlineClassList =
00381         lAirlineClassListList.begin();
00382         itAirlineClassList != lAirlineClassListList.end();
00383         ++itAirlineClassList) {
00384         const stdair::AirlineClassList* lCurrentAirlineClassList_ptr =
00385             *itAirlineClassList;
00386         assert (lCurrentAirlineClassList_ptr != NULL);

```

```

00387
00388     lCorrectAirlineRule = true;
00389     lAtLeastOneDifferentAirline = false;
00390
00391     const stdair::ClassList_StringList_T lClassList_StringList =
00392         lCurrentAirlineClassList_ptr->getAirlineCodeList();
00393
00394     // Compare the segment path airline list with the fare rule airline list.
00395     if (lClassList_StringList.size() == lSegmentPath.size()) {
00396         // If the two sizes are equal, we need to compare the airline codes.
00397         stdair::SegmentPath_T::const_iterator itSegmentPath =
00398             lSegmentPath.begin();
00399
00400         stdair::ClassList_StringList_T::const_iterator itClassList_String =
00401             lClassList_StringList.begin();
00402         // Browse the segment path airline code list (while the segment path
00403         // airline list is equal to the fare rule airline list).
00404         while (itSegmentPath != lSegmentPath.end()
00405             && lAtLeastOneDifferentAirline == false) {
00406
00407             // Get the segment airline code.
00408             const std::string lSegmentDateKey = *itSegmentPath;
00409             const stdair::ParsedKey& lParsedKey =
00410                 stdair::BomKeyManager::extractKeys (lSegmentDateKey);
00411             const stdair::InventoryKey& lInventoryKey =
00412                 lParsedKey.getInventoryKey();
00413             const stdair::AirlineCode_T& lSegmentAirlineCode =
00414                 lInventoryKey.getAirlineCode();
00415
00416             // Get the fare rule airline code.
00417             const stdair::AirlineCode_T& lFareRuleAirlineCode =
00418                 *itClassList_String;
00419
00420             if (lSegmentAirlineCode != lFareRuleAirlineCode) {
00421                 lAtLeastOneDifferentAirline = true;
00422             }
00423             itSegmentPath++;
00424             itClassList_String++;
00425         }
00426     } else {
00427         // If the two sizes are different, the fare rule does not match the
00428         // travel solution into question.
00429         lCorrectAirlineRule = false;
00430     }
00431
00432     // If one segment airline code and one fare rule airline code are
00433     // different then the fare rule does not match the travel solution.
00434     if (lAtLeastOneDifferentAirline == true) {
00435         lCorrectAirlineRule = false;
00436     }
00437
00438     // If the current fare rule is a match, add the fare option structure
00439     // to the travel solution into question.
00440     if (lCorrectAirlineRule == true) {
00441         _atLeastOneAvailableAirlineClassRule = true;
00442         // Get the booking request trip type.
00443         const stdair::TripType_T& lTripType =
00444             iBookingRequest.getTripType();
00445
00446         // Get the travel fare.
00447         stdair::Fare_T lFare =
00448             lCurrentAirlineClassList_ptr->getFare();
00449         // If the trip into question is the inbound or outbound part of a round
00450         trip,
00451         // the applicable fare is a half RT fare.
00452         if (lTripType == "RI" || lTripType == "RO") {
00453             lFare /= 2;
00454         }
00455         // Set the travel fare option.
00456         iFareOption.setFare (lFare);
00457         // Copy the class path list into the fare option.
00458         const stdair::ClassList_StringList_T& lClassCodeList =
00459             lCurrentAirlineClassList_ptr->getClassCodeList();
00460         for (stdair::ClassList_StringList_T::const_iterator itClassCodeList =
00461             lClassCodeList.begin();
00462             itClassCodeList != lClassCodeList.end(); ++itClassCodeList ) {
00463             const stdair::ClassList_String_T& lClassCodeList = *itClassCodeList;
00464             iFareOption.addClassList (lClassCodeList);
00465         }
00466
00467         // Add the fare option to the travel solution into question.
00468         ioTravelSolution.addFareOption (iFareOption);
00469
00470         // DEBUG
00471         STDAIR_LOG_DEBUG ("Segment path: " << lFirstSPParsedKey.toString()
00472             << ". A corresponding fare option for the '")

```

```

00473             << lCurrentAirlineClassList_ptr->describeKey()
00474             << " " class is: " << iFareOption);
00475
00476         iFareOption.emptyClassList();
00477     }
00478 }
00479
00480 }
00481
00482 // //////////////////////////////////////
00483 stdair::ParsedKey FareQuoter::
00484 getFirstSPParsedKey (stdair::TravelSolutionStruct& ioTravelSolution) {
00485
00486     // Get the segment-path of the travel solution.
00487     const stdair::SegmentPath_T& lSegmentPath =
00488         ioTravelSolution.getSegmentPath();
00489
00490     // Get the number of segments of the travel solution.
00491     const stdair::NbOfSegments_T& lNbSegments = lSegmentPath.size();
00492
00493     // Sanity check: there is at least one segment in the travel solution.
00494     assert (lNbSegments >= 1);
00495
00496     // Get the first segment of the travel solution.
00497     const std::string& lFirstSegmentDateKey = lSegmentPath.front();
00498
00499     // Get the parsed key of the first segment of the travel solution.
00500     const stdair::ParsedKey& lFirstSegmentParsedKey =
00501         stdair::BomKeyManager::extractKeys (lFirstSegmentDateKey);
00502
00503     return lFirstSegmentParsedKey;
00504 }
00505
00506 // //////////////////////////////////////
00507 stdair::ParsedKey FareQuoter::
00508 getLastSPParsedKey (stdair::TravelSolutionStruct& ioTravelSolution) {
00509
00510     // Get the segment-path of the travel solution.
00511     const stdair::SegmentPath_T& lSegmentPath =
00512         ioTravelSolution.getSegmentPath();
00513
00514     // Get the number of segments of the travel solution.
00515     const stdair::NbOfSegments_T& lNbSegments = lSegmentPath.size();
00516
00517     // Sanity check: there is at least one segment in the travel solution.
00518     assert (lNbSegments >= 1);
00519
00520     // Get the last segment of the travel solution.
00521     const std::string& lLastSegmentDateKey = lSegmentPath.back();
00522
00523     // Get the parsed key of the last segment of the travel solution.
00524     const stdair::ParsedKey& lLastSegmentParsedKey =
00525         stdair::BomKeyManager::extractKeys (lLastSegmentDateKey);
00526
00527     return lLastSegmentParsedKey;
00528 }
00529
00530 }
00531
00532 // //////////////////////////////////////
00533 void FareQuoter::
00534 displayMissingFareRuleMessage (const stdair::BookingRequestStruct&
00535 iBookingRequest,
00536                             stdair::TravelSolutionStruct& ioTravelSolution
00537 ) {
00538
00539     // Get the origin of the first segment in order to get the origin of
00540     // the solution.
00541     const stdair::ParsedKey lFirstSPParsedKey =
00542         getFirstSPParsedKey(ioTravelSolution);
00543     const stdair::AirportCode_T& lOrigin = lFirstSPParsedKey._boardingPoint;
00544
00545     // Get the destination of the last segment in order to get the
00546     // destination of the solution.
00547     const stdair::ParsedKey& lLastSegmentKey =
00548         getLastSPParsedKey(ioTravelSolution);
00549     const stdair::AirportCode_T& lDestination = lLastSegmentKey._offPoint;
00550
00551     // Construct the Airport pair stream of the segment path.
00552     const stdair::AirportPairKey lAirportPairKey (lOrigin, lDestination);
00553
00554     // Get the date of the first segment date key.
00555     const stdair::FlightDateKey& lFlightDateKey =
00556         lFirstSPParsedKey.getFlightDateKey();
00557
00558     // Get the point-of-sale of the booking request.
00559     const stdair::CityCode_T& lPointOfSale = iBookingRequest.getPOS();

```

```

00558 // Get the booking request channel.
00559 const stdair::ChannelLabel_T& lChannel =
00560     iBookingRequest.getBookingChannel();
00561 // Construct the corresponding POS-channel primary key.
00562 const stdair::PosChannelKey lFarePosChannelKey (lPointOfSale, lChannel);
00563
00564 // Get the booking request date time.
00565 const stdair::DateTime_T& lRequestDateTime =
00566     iBookingRequest.getRequestDateTime();
00567
00568 // If no fare rule has a corresponding date range, the pricing is not
00569 // possible, throw an exception.
00570 if (_atLeastOneAvailableDateRule == false) {
00571     const stdair::SegmentDateKey lSegmentDateKey =
00572         lFirstSPParsedKey.getSegmentKey();
00573     STDAIR_LOG_ERROR ("No available fare rule corresponding to the "
00574         "flight date " << lFlightDateKey.toString()
00575         << " and the Origin-Destination pair: "
00576         << lSegmentDateKey.toString());
00577     throw FlightDateNotFoundException ("No available fare rule for the "
00578         "flight date "
00579         + lFlightDateKey.toString()
00580         + " and the Origin-Destination pair: "
00581         + lSegmentDateKey.toString());
00582 }
00583 // If no fare rule has a corresponding pos channel, the pricing is not
00584 // possible,
00585 // throw an exception.
00586 else if (_atLeastOneAvailablePosChannel == false) {
00587     STDAIR_LOG_ERROR ("No available fare rule corresponding to the "
00588         "point of sale " << lPointOfSale
00589         << ", to the channel " << lChannel
00590         << ", to the flight date "
00591         << lFlightDateKey.toString()
00592         << " and to the Origin-Destination pair: "
00593         << lAirportPairKey.toString());
00594     throw PosOrChannelNotFoundException ("No available fare rule for the "
00595         "point of sale " + lPointOfSale
00596         + ", the channel " + lChannel
00597         + ", the flight date "
00598         + lFlightDateKey.toString()
00599         + " and the Origin-Destination pair: "
00600         + lAirportPairKey.toString());
00601 // If no fare rule has a corresponding time range, the pricing is not
00602 // possible,
00603 // throw an exception.
00604 else if (_atLeastOneAvailableTimeRule == false) {
00605     STDAIR_LOG_ERROR ("No available fare rule corresponding to '"
00606         << lFirstSPParsedKey.toString() << "' (parsed key) and
00607     to '"
00608         << lFarePosChannelKey.toString() << "' (POS and
00609     channel)");
00610     throw FlightTimeNotFoundException ("No available fare rule corresponding
00611     "
00612         "to '" + lFirstSPParsedKey.toString()
00613         + "' (parsed key) and to '"
00614         + lFarePosChannelKey.toString()
00615         + "' (POS and channel)");
00616 }
00617 // If no fare rule matches the advance purchase, trip type and stay
00618 // duration criterion, the pricing is not possible, throw an exception.
00619 else if (_atLeastOneAvailableFeaturesRule == false) {
00620     // Get the stay duration of the booking request.
00621     const stdair::DayDuration_T& lStayDuration=
00622         iBookingRequest.getStayDuration();
00623     std::ostreamstream lStayDurationStream;
00624     lStayDurationStream << lStayDuration;
00625     const std::string lStayDurationString (lStayDurationStream.str());
00626
00627     // Get the booking request trip type.
00628     const stdair::TripType_T& lTripType =
00629         iBookingRequest.getTripType();
00630
00631     STDAIR_LOG_ERROR ("No available fare rule corresponding to a "
00632         "trip type " << lTripType
00633         << ", to a stay duration of " << lStayDurationString
00634         << ", to a request date time of " << lRequestDateTime
00635         << ", to '" << lFirstSPParsedKey.toString()
00636         << "' (parsed key) and to '"
00637         << lFarePosChannelKey << "' (POS and channel)");
00638     throw FeaturesNotFoundException ("No available fare rule corresponding to
00639     a "
00640         "trip type " + lTripType
00641         + ", to a stay duration of "
00642         + lStayDurationString

```

```

00638             + ", to a request date time of "
00639             + boost::posix_time::to_simple_string(
00640         lRequestDateTime)
00641             + ", to '" + lFirstSPParsedKey.toString()
00642             + "' (parsed key) and to '"
00643             + lFarePosChannelKey.toString()
00644             + "' (POS and channel)");
00645     }
00646     assert (_atLeastOneAvailableAirlineClassRule == false);
00647     // If no fare rule matches the airline class path, the pricing is not
00648     // possible, throw an exception.
00649     STDAIR_LOG_ERROR ("No available fare rule corresponding to '"
00650         << lFirstSPParsedKey.toString() << "' (parsed key), to '"
00651         << iBookingRequest.describe()
00652         << "' (booking request) and to '"
00653         << lFarePosChannelKey.toString() << "' (POS and channel)"
00654     );
00655     throw AirlineNotFoundException ("No available fare rule corresponding to '"
00656         + lFirstSPParsedKey.toString()
00657         + "' (parsed key), to '"
00658         + iBookingRequest.describe()
00659         + "' (booking request) and to '"
00660         + lFarePosChannelKey.toString()
00661         + "' (POS and channel)");
00662 }

```

23.37 simfq/command/FareQuoter.hpp File Reference

```

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

```

Classes

- class [SIMFQT::FareQuoter](#)
Command wrapping the pricing request process.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [SIMFQT](#)

23.38 FareQuoter.hpp

```

00001 #ifndef __SIMFQT_CMD_FAREQUOTER_HPP
00002 #define __SIMFQT_CMD_FAREQUOTER_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/TravelSolutionTypes.hpp>
00010
00011 namespace stdair {
00012     class BomRoot;
00013     struct BookingRequestStruct;
00014     struct TravelSolutionStruct;
00015     struct ParsedKey;
00016     class AirportPair;
00017     class PosChannel;
00018     class DatePeriod;
00019     class TimePeriod;
00020     class FareFeatures;
00021 }
00022
00023 namespace SIMFQT {

```

```

00025
00029 class FareQuoter {
00032     friend class SIMFQT_Service;
00033
00034 private:
00035     // ////////////////////////////////// Business support methods //////////////////////////////////
00045     static void priceQuote (const stdair::BookingRequestStruct&,
00046                             stdair::TravelSolutionList_T&,
00047                             const stdair::BomRoot&);
00048
00060     static void priceQuote (const stdair::BookingRequestStruct&,
00061                             stdair::TravelSolutionStruct&,
00062                             const stdair::BomRoot&);
00063
00074     static void priceQuote (const stdair::BookingRequestStruct&,
00075                             stdair::TravelSolutionStruct&,
00076                             const stdair::AirportPair&);
00077
00092     static void priceQuote (const stdair::BookingRequestStruct&,
00093                             stdair::TravelSolutionStruct&,
00094                             const stdair::DatePeriod&,
00095                             const stdair::AirportPair&);
00096
00108     static void priceQuote (const stdair::BookingRequestStruct&,
00109                             stdair::TravelSolutionStruct&,
00110                             const stdair::PosChannel&);
00111
00126     static void priceQuote (const stdair::BookingRequestStruct&,
00127                             stdair::TravelSolutionStruct&,
00128                             const stdair::TimePeriod&,
00129                             const stdair::PosChannel&);
00130
00148     static void priceQuote (const stdair::BookingRequestStruct&,
00149                             stdair::TravelSolutionStruct&,
00150                             const stdair::FareFeatures&,
00151                             const stdair::PosChannel&,
00152                             stdair::FareOptionStruct&);
00153
00157     static void reset ();
00158
00168     static void displayMissingFareRuleMessage (const
stdair::BookingRequestStruct&,
00169                                                stdair::TravelSolutionStruct&);
00170
00178     static stdair::ParsedKey getFirstSPParsedKey (stdair::TravelSolutionStruct&
);
00179
00187     static stdair::ParsedKey getLastSPParsedKey (stdair::TravelSolutionStruct&)
;
00188
00189
00190
00191 private:
00192     // ////////////////////////////////// Construction and destruction //////////////////////////////////
00196     FareQuoter();
00197
00201     FareQuoter(const FareQuoter&);
00202
00206     ~FareQuoter();
00207
00208 private:
00209
00212     static bool _atLeastOneAvailableDateRule;
00213
00216     static bool _atLeastOneAvailablePosChannel;
00217
00221     static bool _atLeastOneAvailableTimeRule;
00222
00226     static bool _atLeastOneAvailableFeaturesRule;
00227
00231     static bool _atLeastOneAvailableAirlineClassRule;
00232
00233 };
00234
00235 }
00236 #endif // __SIMFQT_CMD_FAREQUOTER_HPP
00237

```

23.39 simfq/command/FareRuleGenerator.cpp File Reference

```
#include <cassert>
```



```

#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/FareFeatures.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <simfqt/bom/FareRuleStruct.hpp>
#include <simfqt/command/FareRuleGenerator.hpp>

```

Namespaces

- namespace [SIMFQT](#)

23.40 FareRuleGenerator.cpp

```

00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/bom/BomManager.hpp>
00008 #include <stdair/bom/BomRoot.hpp>
00009 #include <stdair/factory/FacBomManager.hpp>
00010 #include <stdair/service/Logger.hpp>
00011 #include <stdair/bom/AirportPair.hpp>
00012 #include <stdair/bom/PosChannel.hpp>
00013 #include <stdair/bom/DatePeriod.hpp>
00014 #include <stdair/bom/TimePeriod.hpp>
00015 #include <stdair/bom/FareFeatures.hpp>
00016 #include <stdair/bom/AirlineClassList.hpp>
00017 // SimFQT
00018 #include <simfqt/bom/FareRuleStruct.hpp>
00019 #include <simfqt/command/FareRuleGenerator.hpp>
00020 >
00021 namespace SIMFQT {
00022
00023 // //////////////////////////////////////
00024 void FareRuleGenerator::
00025 createAirportPair (stdair::BomRoot& ioBomRoot,
00026                   const FareRuleStruct& iFareRuleStruct) {
00027
00028     // Create the airport-pair primary key.
00029     const stdair::AirportCode_T& lBoardPoint = iFareRuleStruct.getOrigin ();
00030     const stdair::AirportCode_T& lOffPoint =
00031         iFareRuleStruct.getDestination ();
00032     const stdair::AirportPairKey lAirportPairKey (lBoardPoint, lOffPoint);
00033
00034     // Check that the airport-pair object is not already existing. If an
00035     // airport-pair object with the same key has not already been created,
00036     // create it and link it to the ioBomRoot object.
00037     stdair::AirportPair* lAirportPair_ptr = stdair::BomManager::
00038         getObjectPtr<stdair::AirportPair> (ioBomRoot, lAirportPairKey.toString());
00039
00040     if (lAirportPair_ptr == NULL) {
00041         lAirportPair_ptr =
00042             &stdair::FacBom<stdair::AirportPair>::instance().
00043             create (lAirportPairKey);
00044         stdair::FacBomManager::addToListAndMap (ioBomRoot, *lAirportPair_ptr);
00045         stdair::FacBomManager::linkWithParent (ioBomRoot, *lAirportPair_ptr);
00046     }
00047     // Sanity check.
00048     assert (lAirportPair_ptr != NULL);
00049
00050     stdair::AirportPair& lAirportPair = *lAirportPair_ptr;
00051     // Generate the date-period object corresponding to the given
00052     // fareRule.
00053     createDateRange (lAirportPair, iFareRuleStruct);
00054 }

```

```

00054 }
00055
00056 // //////////////////////////////////////
00057 void FareRuleGenerator::
00058 createDateRange (stdair::AirportPair& iAirportPair,
00059                 const FareRuleStruct& iFareRuleStruct) {
00060
00061     // Create the fare date-period primary key.
00062     const stdair::Date_T& lDateRangeStart =
00063         iFareRuleStruct.getDateRangeStart ();
00064     const stdair::Date_T& lDateRangeEnd =
00065         iFareRuleStruct.getDateRangeEnd ();
00066     const stdair::DatePeriod_T lDatePeriod (lDateRangeStart, lDateRangeEnd);
00067     const stdair::DatePeriodKey lFareDatePeriodKey (lDatePeriod);
00068
00069     // Check that the date-period object is not already existing.
00070     // If a date-period object with the same key has not already been
00071     // created, create it and link it to the airport-pair object.
00072     stdair::DatePeriod* lFareDatePeriod_ptr = stdair::BomManager::
00073         getObjectPtr<stdair::DatePeriod> (iAirportPair,
00074                                           lFareDatePeriodKey.toString());
00075     if (lFareDatePeriod_ptr == NULL) {
00076         lFareDatePeriod_ptr = &stdair::FacBom<stdair::DatePeriod>::instance().
00077             create (lFareDatePeriodKey);
00078         stdair::FacBomManager::addToListAndMap (iAirportPair,
00079                                                 *lFareDatePeriod_ptr);
00080         stdair::FacBomManager::linkWithParent (iAirportPair,
00081                                                 *lFareDatePeriod_ptr);
00082     }
00083     // Sanity check.
00084     assert (lFareDatePeriod_ptr != NULL);
00085
00086     stdair::DatePeriod& lDateRange = *lFareDatePeriod_ptr;
00087     // Generate the point_of_sale-channel object corresponding to
00088     // the given fareRule.
00089     createPOSChannel (lDateRange, iFareRuleStruct);
00090 }
00091
00092 // //////////////////////////////////////
00093 void FareRuleGenerator::
00094 createPOSChannel (stdair::DatePeriod& iDatePeriod,
00095                 const FareRuleStruct& iFareRuleStruct) {
00096
00097     // Create the point-of-sale-channel primary key.
00098     const stdair::CityCode_T& lPosition = iFareRuleStruct.getPOS ();
00099     const stdair::ChannelLabel_T& lChannel =
00100         iFareRuleStruct.getChannel ();
00101     const stdair::PosChannelKey lFarePosChannelKey (lPosition, lChannel);
00102
00103     // Check that the point_of_sale-channel object is not already existing.
00104     // If a point_of_sale-channel object with the same key has not already
00105     // been created, create it and link it to the date-period object.
00106     stdair::PosChannel* lFarePosChannel_ptr = stdair::BomManager::
00107         getObjectPtr<stdair::PosChannel> (iDatePeriod,
00108                                           lFarePosChannelKey.toString());
00109     if (lFarePosChannel_ptr == NULL) {
00110         lFarePosChannel_ptr = &stdair::FacBom<stdair::PosChannel>::instance().
00111             create (lFarePosChannelKey);
00112         stdair::FacBomManager::addToListAndMap (iDatePeriod,
00113                                                 *lFarePosChannel_ptr);
00114         stdair::FacBomManager::linkWithParent (iDatePeriod,
00115                                                 *lFarePosChannel_ptr);
00116     }
00117     // Sanity check.
00118     assert (lFarePosChannel_ptr != NULL);
00119
00120     stdair::PosChannel& lPosChannel = *lFarePosChannel_ptr;
00121     // Generate the time-period object corresponding to the given
00122     // fareRule.
00123     createTimeRange (lPosChannel, iFareRuleStruct);
00124 }
00125
00126 // //////////////////////////////////////
00127 void FareRuleGenerator::
00128 createTimeRange (stdair::PosChannel& iPosChannel,
00129                 const FareRuleStruct& iFareRuleStruct) {
00130
00131     // Create the fare time-period primary key.
00132     const stdair::Time_T& lTimeRangeStart =
00133         iFareRuleStruct.getTimeRangeStart ();
00134     const stdair::Time_T& lTimeRangeEnd =
00135         iFareRuleStruct.getTimeRangeEnd ();
00136     const stdair::TimePeriodKey lFareTimePeriodKey (lTimeRangeStart,
00137                                                       lTimeRangeEnd);
00140

```

```

00141
00142 // Check that the time-period object is not already existing.
00143 // If a time-period object with the same key has not already been
00144 // created, create it and link it to the point_of_sale-channel object.

00145 stdair::TimePeriod* lFareTimePeriod_ptr = stdair::BomManager::
00146     getObjectPtr<stdair::TimePeriod> (iPosChannel,
00147         lFareTimePeriodKey.toString());
00148 if (lFareTimePeriod_ptr == NULL) {
00149     lFareTimePeriod_ptr = &stdair::FacBom<stdair::TimePeriod>::instance().
00150         create (lFareTimePeriodKey);
00151     stdair::FacBomManager::addToListAndMap (iPosChannel,
00152         *lFareTimePeriod_ptr);
00153     stdair::FacBomManager::linkWithParent (iPosChannel,
00154         *lFareTimePeriod_ptr);
00155 }
00156 // Sanity check.
00157 assert (lFareTimePeriod_ptr != NULL);
00158
00159 stdair::TimePeriod* lTimeRange = *lFareTimePeriod_ptr;
00160 // Generate the fare-features object corresponding to the given
00161 // fareRule.
00162 createFareFeatures (lTimeRange, iFareRuleStruct);
00163
00164 }
00165
00166 // //////////////////////////////////////
00167 void FareRuleGenerator::
00168 createFareFeatures (stdair::TimePeriod& iTimePeriod,
00169     const FareRuleStruct& iFareRuleStruct) {
00170
00171     // Create the fare-features primary key.
00172     const stdair::TripType_T& lTripType =
00173         iFareRuleStruct.getTripType ();
00174     const stdair::DayDuration_T& lAdvancePurchase =
00175         iFareRuleStruct.getAdvancePurchase ();
00176     const stdair::SaturdayStay_T& lSaturdayStay =
00177         iFareRuleStruct.getSaturdayStay ();
00178     const stdair::ChangeFees_T& lChangeFees =
00179         iFareRuleStruct.getChangeFees ();
00180     const stdair::NonRefundable_T& lNonRefundable =
00181         iFareRuleStruct.getNonRefundable ();
00182     const stdair::DayDuration_T& lMinimumStay =
00183         iFareRuleStruct.getMinimumStay ();
00184     const stdair::FareFeaturesKey
00185         lFareFeaturesKey (lTripType, lAdvancePurchase, lSaturdayStay,
00186             lChangeFees, lNonRefundable, lMinimumStay);
00187
00188     // Check that the fare features object is not already existing.
00189     // If a fare features object with the same key has not already been
00190     // created, create it and link it to the time-period object.
00191     stdair::FareFeatures* lFareFeatures_ptr = stdair::BomManager::
00192         getObjectPtr<stdair::FareFeatures> (iTimePeriod,
00193             lFareFeaturesKey.toString());
00194     if (lFareFeatures_ptr == NULL) {
00195         lFareFeatures_ptr = &stdair::FacBom<stdair::FareFeatures>::instance().
00196             create (lFareFeaturesKey);
00197         assert (lFareFeatures_ptr != NULL);
00198         stdair::FacBomManager::addToListAndMap (iTimePeriod,
00199             *lFareFeatures_ptr);
00200         stdair::FacBomManager::linkWithParent (iTimePeriod,
00201             *lFareFeatures_ptr);
00202     }
00203     // Sanity check.
00204     assert (lFareFeatures_ptr != NULL);
00205
00206     stdair::FareFeatures& lFareFeatures = *lFareFeatures_ptr;
00207     // Generate the airline-class list object corresponding to the
00208     // given fareRule
00209     createAirlineClassList (lFareFeatures, iFareRuleStruct);
00210
00211 }
00212
00213 // //////////////////////////////////////
00214 void FareRuleGenerator::
00215 createAirlineClassList (stdair::FareFeatures& iFareFeatures,
00216     const FareRuleStruct& iFareRuleStruct) {
00217
00218     // Create the AirlineClassList primary key.
00219     const unsigned int lAirlineListSize =
00220         iFareRuleStruct.getAirlineListSize();
00221     const unsigned int lClassCodeListSize =
00222         iFareRuleStruct.getClassCodeListSize();
00223     assert (lAirlineListSize == lClassCodeListSize);
00224     const stdair::AirlineClassListKey
00225         lAirlineClassListKey (iFareRuleStruct.getAirlineList(),
00226             iFareRuleStruct.getClassCodeList());

```

```

00227     const stdair::Fare_T& lFare = iFareRuleStruct.getFare ();
00228
00229     // Create the airline class list object and link it to the fare features
00230     // object.
00231     stdair::AirlineClassList* lAirlineClassList_ptr =
00232         &stdair::FacBom<stdair::AirlineClassList>::instance().
00233         create (lAirlineClassListKey);
00234     lAirlineClassList_ptr->setFare(lFare);
00235     stdair::FacBomManager::addToListAndMap (iFareFeatures,
00236         *lAirlineClassList_ptr);
00237     stdair::FacBomManager::linkWithParent (iFareFeatures,
00238         *lAirlineClassList_ptr);
00239 }
00240
00241 }
00242

```

23.41 simfqt/command/FareRuleGenerator.hpp File Reference

```

#include <stdair/command/CmdAbstract.hpp>
#include <simfqt/SIMFQT_Types.hpp>

```

Classes

- class [SIMFQT::FareRuleGenerator](#)

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [SIMFQT](#)
- namespace [SIMFQT::FareParserHelper](#)

23.42 FareRuleGenerator.hpp

```

00001 #ifndef __SIMFQT_CMD_FARERULEGENERATOR_HPP
00002 #define __SIMFQT_CMD_FARERULEGENERATOR_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // StdAir
00008 #include <stdair/command/CmdAbstract.hpp>
00009 // Simfqt
00010 #include <simfqt/SIMFQT_Types.hpp>
00011
00012 // Forward declarations
00013 namespace stdair {
00014     class BomRoot;
00015     class FareRule;
00016     class AirportPair;
00017     class DatePeriod;
00018     class PosChannel;
00019     class TimePeriod;
00020     class FareFeatures;
00021     class AirlineClassList;
00022 }
00023
00024 namespace SIMFQT {
00025
00026     // Forward declarations
00027     struct FareRuleStruct;
00028     namespace FareParserHelper {
00029         struct doEndFare;
00030     }
00031
00032     class FareRuleGenerator : public stdair::CmdAbstract {
00033
00034     public:
00035         // Only the following class may use methods of FareGenerator.
00036         // Indeed, as those methods build the BOM, it is not good to expose
00037         // them public.

```

```

00038     friend class FareFileParser;
00039     friend struct FareParserHelper::doEndFare;
00040     friend class FareParser;
00041
00042 private:
00043
00052     static void createAirportPair (stdair::BomRoot&,
00053                                   const FareRuleStruct&);
00054
00063     static void createDateRange (stdair::AirportPair&,
00064                                  const FareRuleStruct&);
00065
00074     static void createPOSChannel (stdair::DatePeriod&,
00075                                  const FareRuleStruct&);
00076
00085     static void createTimeRange (stdair::PosChannel&,
00086                                  const FareRuleStruct&);
00087
00096     static void createFareFeatures (stdair::TimePeriod&,
00097                                    const FareRuleStruct&);
00098
00107     static void createAirlineClassList (stdair::FareFeatures&,
00108                                         const FareRuleStruct&);
00109
00110
00111 };
00112 };
00113
00114 }
00115 #endif // __SIMFQT_CMD_FARERULEGENERATOR_HPP

```

23.43 simfqt/config/simfqt-paths.hpp File Reference

Macros

- #define [PACKAGE](#) "simfqt"
- #define [PACKAGE_NAME](#) "SIMFQT"
- #define [PACKAGE_VERSION](#) "0.1.3"
- #define [PREFIXDIR](#) "/usr"
- #define [EXEC_PREFIX](#) "/usr"
- #define [BINDIR](#) "/usr/bin"
- #define [LIBDIR](#) "/usr/lib"
- #define [LIBEXECDIR](#) "/usr/libexec"
- #define [SBINDIR](#) "/usr/sbin"
- #define [SYSCONFDIR](#) "/usr/etc"
- #define [INCLUDEDIR](#) "/usr/include"
- #define [DATAROOTDIR](#) "/usr/share"
- #define [DATADIR](#) "/usr/share"
- #define [DOCDIR](#) "/usr/share/doc/simfqt-0.1.3"
- #define [MANDIR](#) "/usr/share/man"
- #define [INFODIR](#) "/usr/share/info"
- #define [HTMLDIR](#) "/usr/share/doc/simfqt-0.1.3/html"
- #define [PDFDIR](#) "/usr/share/doc/simfqt-0.1.3/html"
- #define [STDAIR_SAMPLE_DIR](#) "/usr/share/stdair/samples"

23.43.1 Macro Definition Documentation

23.43.1.1 #define PACKAGE "simfqt"

Definition at line 4 of file [simfqt-paths.hpp](#).

23.43.1.2 #define PACKAGE_NAME "SIMFQT"

Definition at line 5 of file [simfqt-paths.hpp](#).

Referenced by [readConfiguration\(\)](#).

23.43.1.3 `#define PACKAGE_VERSION "0.1.3"`

Definition at line 6 of file [simfqt-paths.hpp](#).

Referenced by [readConfiguration\(\)](#).

23.43.1.4 `#define PREFIXDIR "/usr"`

Definition at line 7 of file [simfqt-paths.hpp](#).

Referenced by [readConfiguration\(\)](#).

23.43.1.5 `#define EXEC_PREFIX "/usr"`

Definition at line 8 of file [simfqt-paths.hpp](#).

23.43.1.6 `#define BINDIR "/usr/bin"`

Definition at line 9 of file [simfqt-paths.hpp](#).

23.43.1.7 `#define LIBDIR "/usr/lib"`

Definition at line 10 of file [simfqt-paths.hpp](#).

23.43.1.8 `#define LIBEXECDIR "/usr/libexec"`

Definition at line 11 of file [simfqt-paths.hpp](#).

23.43.1.9 `#define SBINDIR "/usr/sbin"`

Definition at line 12 of file [simfqt-paths.hpp](#).

23.43.1.10 `#define SYSCONFDIR "/usr/etc"`

Definition at line 13 of file [simfqt-paths.hpp](#).

23.43.1.11 `#define INCLUDEDIR "/usr/include"`

Definition at line 14 of file [simfqt-paths.hpp](#).

23.43.1.12 `#define DATAROOTDIR "/usr/share"`

Definition at line 15 of file [simfqt-paths.hpp](#).

23.43.1.13 `#define DATADIR "/usr/share"`

Definition at line 16 of file [simfqt-paths.hpp](#).

23.43.1.14 `#define DOCDIR "/usr/share/doc/simfqt-0.1.3"`

Definition at line 17 of file [simfqt-paths.hpp](#).

23.43.1.15 `#define MANDIR "/usr/share/man"`

Definition at line 18 of file [simfqt-paths.hpp](#).

23.43.1.16 `#define INFODIR "/usr/share/info"`

Definition at line 19 of file [simfqt-paths.hpp](#).

23.43.1.17 `#define HTMLDIR "/usr/share/doc/simfqt-0.1.3/html"`

Definition at line 20 of file [simfqt-paths.hpp](#).

23.43.1.18 #define PDFDIR "/usr/share/doc/simfqt-0.1.3/html"

Definition at line 21 of file [simfqt-paths.hpp](#).

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

Definition at line 22 of file [simfqt-paths.hpp](#).

23.44 simfqt-paths.hpp

```
00001 #ifndef __SIMFQT_PATHS_HPP__
00002 #define __SIMFQT_PATHS_HPP__
00003
00004 #define PACKAGE "simfqt"
00005 #define PACKAGE_NAME "SIMFQT"
00006 #define PACKAGE_VERSION "0.1.3"
00007 #define PREFIXDIR "/usr"
00008 #define EXEC_PREFIX "/usr"
00009 #define BINDIR "/usr/bin"
00010 #define LIBDIR "/usr/lib"
00011 #define LIBEXECDIR "/usr/libexec"
00012 #define SBINDIR "/usr/sbin"
00013 #define SYSCONFDIR "/usr/etc"
00014 #define INCLUDEDIR "/usr/include"
00015 #define DATAROOTDIR "/usr/share"
00016 #define DATADIR "/usr/share"
00017 #define DOCDIR "/usr/share/doc/simfqt-0.1.3"
00018 #define MANDIR "/usr/share/man"
00019 #define INFODIR "/usr/share/info"
00020 #define HTMLDIR "/usr/share/doc/simfqt-0.1.3/html"
00021 #define PDFDIR "/usr/share/doc/simfqt-0.1.3/html"
00022 #define STDAIR_SAMPLE_DIR "/usr/share/stdair/samples"
00023
00024 #endif // __SIMFQT_PATHS_HPP__
```

23.45 simfqt/factory/FacSimfqtServiceContext.cpp File Reference

```
#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <simfqt/factory/FacSimfqtServiceContext.hpp>
#include <simfqt/service/SIMFQT_ServiceContext.hpp>
```

Namespaces

- namespace [SIMFQT](#)

23.46 FacSimfqtServiceContext.cpp

```
00001 // //////////////////////////////////////
00002 // Import section
00003 // //////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/service/FacSupervisor.hpp>
00008 // SimFQT
00009 #include <simfqt/factory/FacSimfqtServiceContext.hpp>
00010 #include <simfqt/service/SIMFQT_ServiceContext.hpp>
00011
00012 namespace SIMFQT {
00013
00014     FacSimfqtServiceContext* FacSimfqtServiceContext::_instance = NULL;
00015
00016     // //////////////////////////////////////
00017     FacSimfqtServiceContext::~FacSimfqtServiceContext
00018     () {
00019         _instance = NULL;
```

```

00019     }
00020
00021     // //////////////////////////////////////
00022     FacSimfqtServiceContext&
FacSimfqtServiceContext::instance() {
00023
00024         if (_instance == NULL) {
00025             _instance = new FacSimfqtServiceContext();
00026             assert (_instance != NULL);
00027
00028             stdair::FacSupervisor::instance().
registerServiceFactory (_instance);
00029         }
00030         return *_instance;
00031     }
00032
00033     // //////////////////////////////////////
00034     SIMFQT_ServiceContext& FacSimfqtServiceContext::create
() {
00035         SIMFQT_ServiceContext* aServiceContext_ptr = NULL;
00036
00037         aServiceContext_ptr = new SIMFQT_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 }

```

23.47 simfqt/factory/FacSimfqtServiceContext.hpp File Reference

```

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

```

Classes

- class [SIMFQT::FacSimfqtServiceContext](#)
Factory for the service context.

Namespaces

- namespace [SIMFQT](#)

23.48 FacSimfqtServiceContext.hpp

```

00001 #ifndef __SIMFQT_FAC_FACSIMFQTSERVICECONTEXT_HPP
00002 #define __SIMFQT_FAC_FACSIMFQTSERVICECONTEXT_HPP
00003
00004 // //////////////////////////////////////
00005 // Import section
00006 // //////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/service/FacServiceAbstract.hpp>
00012
00013 namespace SIMFQT {
00014
00015     class SIMFQT_ServiceContext;
00016
00017     class FacSimfqtServiceContext : public
stdair::FacServiceAbstract {
00018     public:
00019
00020         static FacSimfqtServiceContext& instance();
00021
00022     };
00023
00024 }
00025
00026
00027
00028
00029
00030
00031
00032
00033
00034
00035
00036
00037
00038
00039
00040
00041
00042
00043
00044
00045
00046
00047
00048
00049
00050
00051
00052
00053
00054
00055
00056
00057
00058
00059
00060
00061
00062
00063
00064
00065
00066
00067
00068
00069
00070
00071
00072
00073
00074
00075
00076
00077
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999

```



```

00032
00039     ~FacSimfqtServiceContext();
00040
00048     SIMFQT_ServiceContext& create();
00049
00050
00051 protected:
00057     FacSimfqtServiceContext() {}
00058
00059
00060 private:
00064     static FacSimfqtServiceContext* _instance;
00065 };
00066
00067 }
00068 #endif // __SIMFQT_FAC_FACSIMFQTSERVICECONTEXT_HPP

```

23.49 simfqt/service/SIMFQT_Service.cpp File Reference

```

#include <cassert>
#include <boost/make_shared.hpp>
#include <stdair/basic/BasChronometer.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <simfqt/basic/BasConst_SIMFQT_Service.hpp>
#include <simfqt/factory/FacSimfqtServiceContext.hpp>
#include <simfqt/command/FareParser.hpp>
#include <simfqt/command/FareQuoter.hpp>
#include <simfqt/service/SIMFQT_ServiceContext.hpp>
#include <simfqt/SIMFQT_Service.hpp>

```

Namespaces

- namespace [SIMFQT](#)

23.50 SIMFQT_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/bom/TravelSolutionStruct.hpp>
00012 #include <stdair/bom/BookingRequestStruct.hpp>
00013 #include <stdair/service/Logger.hpp>
00014 #include <stdair/STDAIR_Service.hpp>
00015 // Simfqt
00016 #include <simfqt/basic/BasConst_SIMFQT_Service.hpp>
00017 #include <simfqt/factory/FacSimfqtServiceContext.hpp>
00018 #include <simfqt/command/FareParser.hpp>
00019 #include <simfqt/command/FareQuoter.hpp>
00020 #include <simfqt/service/SIMFQT_ServiceContext.hpp>
00021 #include <simfqt/SIMFQT_Service.hpp>
00022
00023 namespace SIMFQT {
00024
00025     // //////////////////////////////////////
00026     SIMFQT_Service::SIMFQT_Service() : _simfqtServiceContext (NULL) {
00027         assert (false);

```

```

00028     }
00029
00030     // //////////////////////////////////////
00031     SIMFQT_Service::SIMFQT_Service (const SIMFQT_Service& iService) {
00032         assert (false);
00033     }
00034
00035     // //////////////////////////////////////
00036     SIMFQT_Service::SIMFQT_Service (const stdair::BasLogParams& iLogParams)
00037         : _simfqtServiceContext (NULL) {
00038
00039         // Initialise the STDAIR service handler
00040         stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00041             initStdAirService (iLogParams);
00042
00043         // Initialise the service context
00044         initServiceContext();
00045
00046         // Add the StdAir service context to the SIMFQT service context
00047         // \note SIMFQT owns the STDAIR service resources here.
00048         const bool ownStdairService = true;
00049         addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00050
00051         // Initialise the (remaining of the) context
00052         initSimfqtService();
00053     }
00054
00055     // //////////////////////////////////////
00056     SIMFQT_Service::SIMFQT_Service (const stdair::BasLogParams& iLogParams,
00057                                     const stdair::BasDBParams& iDBParams)
00058         : _simfqtServiceContext (NULL) {
00059
00060         // Initialise the STDAIR service handler
00061         stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00062             initStdAirService (iLogParams, iDBParams);
00063
00064         // Initialise the service context
00065         initServiceContext();
00066
00067         // Add the StdAir service context to the SIMFQT service context
00068         // \note SIMFQT owns the STDAIR service resources here.
00069         const bool ownStdairService = true;
00070         addStdAirService (lSTDAIR_Service_ptr, ownStdairService);
00071
00072         // Initialise the (remaining of the) context
00073         initSimfqtService();
00074     }
00075
00076     // //////////////////////////////////////
00077     SIMFQT_Service::
00078     SIMFQT_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr)
00079         : _simfqtServiceContext (NULL) {
00080
00081         // Initialise the service context
00082         initServiceContext();
00083
00084         // Store the STDAIR service object within the (SIMFQT) service context
00085         // \note Simfqt does not own the STDAIR service resources here.
00086         const bool doesNotOwnStdairService = false;
00087         addStdAirService (ioSTDAIR_Service_ptr, doesNotOwnStdairService);
00088
00089         // Initialise the context
00090         initSimfqtService();
00091     }
00092
00093     // //////////////////////////////////////
00094     SIMFQT_Service::~SIMFQT_Service() {
00095         // Delete/Clean all the objects from memory
00096         finalise();
00097     }
00098
00099     // //////////////////////////////////////
00100     void SIMFQT_Service::finalise() {
00101         assert (_simfqtServiceContext != NULL);
00102         // Reset the (Boost.)Smart pointer pointing on the STDAIR_Service object.
00103         _simfqtServiceContext->reset();
00104     }
00105
00106     // //////////////////////////////////////
00107     void SIMFQT_Service::initServiceContext() {
00108         // Initialise the service context
00109         SIMFQT_ServiceContext& lSIMFQT_ServiceContext =
00110             FacSimfqtServiceContext::instance().
00111             create();
00112         _simfqtServiceContext = &lSIMFQT_ServiceContext;
00113     }

```

```

00114 // //////////////////////////////////////
00115 void SIMFQT_Service::
00116 addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_Service_ptr,
00117                  const bool iOwnStdairService) {
00118
00119     // Retrieve the SimFQT service context
00120     assert (_simfqtServiceContext != NULL);
00121     SIMFQT_ServiceContext& lSIMFQT_ServiceContext = *_simfqtServiceContext;
00122
00123     // Store the STDAIR service object within the (SimFQT) service context
00124     lSIMFQT_ServiceContext.setSTDAIR_Service (ioSTDAIR_Service_ptr,
00125                                              iOwnStdairService);
00126 }
00127
00128 // //////////////////////////////////////
00129 stdair::STDAIR_ServicePtr_T SIMFQT_Service::
00130 initStdAirService (const stdair::BasLogParams& iLogParams,
00131                  const stdair::BasDBParams& iDBParams) {
00132
00133     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00134         boost::make_shared<stdair::STDAIR_Service> (iLogParams, iDBParams);
00135     assert (lSTDAIR_Service_ptr != NULL);
00136
00137     return lSTDAIR_Service_ptr;
00138 }
00139
00140 // //////////////////////////////////////
00141 stdair::STDAIR_ServicePtr_T SIMFQT_Service::
00142 initStdAirService (const stdair::BasLogParams& iLogParams) {
00143
00144     stdair::STDAIR_ServicePtr_T lSTDAIR_Service_ptr =
00145         boost::make_shared<stdair::STDAIR_Service> (iLogParams);
00146     assert (lSTDAIR_Service_ptr != NULL);
00147
00148     return lSTDAIR_Service_ptr;
00149 }
00150
00151 // //////////////////////////////////////
00152 void SIMFQT_Service::initSimfqtService() {
00153     // Do nothing at this stage. A sample BOM tree may be built by
00154     // calling the buildSampleBom() method
00155 }
00156
00157 // //////////////////////////////////////
00158 void SIMFQT_Service::
00159 parseAndLoad (const FareFilePath& iFareFilename) {
00160
00161     // Retrieve the BOM root object.
00162     assert (_simfqtServiceContext != NULL);
00163     SIMFQT_ServiceContext& lSIMFQT_ServiceContext = *
00164 _simfqtServiceContext;
00165     stdair::STDAIR_Service& lSTDAIR_Service =
00166         lSIMFQT_ServiceContext.getSTDAIR_Service();
00167     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00168
00169     // Initialise the airline inventories
00170     FareParser::fareRuleGeneration (iFareFilename
00171 , lBomRoot);
00172 }
00173
00174 // //////////////////////////////////////
00175 void SIMFQT_Service::buildSampleBom() {
00176
00177     // Retrieve the SimFQT service context
00178     if (_simfqtServiceContext == NULL) {
00179         throw stdair::NonInitialisedServiceException ("The SimFQT service "
00180 "has not been initialised")
00181 ;
00182     }
00183     assert (_simfqtServiceContext != NULL);
00184
00185     // Retrieve the SimFQT service context and whether it owns the Stdair
00186     // service
00187     SIMFQT_ServiceContext& lSIMFQT_ServiceContext = *
00188 _simfqtServiceContext;
00189     const bool doesOwnStdairService =
00190         lSIMFQT_ServiceContext.getOwnStdairServiceFlag();
00191
00192     // Retrieve the StdAir service object from the (SimFQT) service context
00193     stdair::STDAIR_Service& lSTDAIR_Service =
00194         lSIMFQT_ServiceContext.getSTDAIR_Service();
00195
00196     if (doesOwnStdairService == true) {
00197         //
00198         lSTDAIR_Service.buildSampleBom();
00199     }
00200 }

```

```

00228     }
00229
00230     // ////////////////////////////////////////
00231     stdair::BookingRequestStruct SIMFQT_Service::buildBookingRequest
(const bool isForCRS) {
00232
00233         // Retrieve the SIMFQT service context
00234         if (_simfqtServiceContext == NULL) {
00235             throw stdair::NonInitialisedServiceException ("The Simfqt service has not
"
00236                                                         "been initialised");
00237         }
00238         assert (_simfqtServiceContext != NULL);
00239
00240         SIMFQT_ServiceContext& lSIMFQT_ServiceContext = *
_simfqtServiceContext;
00241
00242         // Retrieve the STDAIR service object from the (Simfqt) service context
00243         stdair::STDAIR_Service& lSTDAIR_Service =
lSIMFQT_ServiceContext.getSTDAIR_Service();
00244
00245         // Delegate the BOM building to the dedicated service
00246         stdair::BookingRequestStruct oBookingRequest =
lSTDAIR_Service.buildSampleBookingRequest (isForCRS);
00247
00248         return oBookingRequest;
00249     }
00250
00251     // ////////////////////////////////////////
00252     void SIMFQT_Service::
00253     buildSampleTravelSolutions(
stdair::TravelSolutionList_T& ioTravelSolutionList){
00254
00255         // Retrieve the SIMFQT service context
00256         if (_simfqtServiceContext == NULL) {
00257             throw stdair::NonInitialisedServiceException ("The Simfqt service has not
"
00258                                                         "been initialised");
00259         }
00260         assert (_simfqtServiceContext != NULL);
00261
00262         SIMFQT_ServiceContext& lSIMFQT_ServiceContext = *
_simfqtServiceContext;
00263
00264         // Retrieve the STDAIR service object from the (Simfqt) service context
00265         stdair::STDAIR_Service& lSTDAIR_Service =
lSIMFQT_ServiceContext.getSTDAIR_Service();
00266
00267         // Delegate the BOM building to the dedicated service
00268         lSTDAIR_Service.buildSampleTravelSolutionForPricing (ioTravelSolutionList);
00269     }
00270
00271     // ////////////////////////////////////////
00272     std::string SIMFQT_Service::csvDisplay() const {
00273
00274         // Retrieve the SIMFQT service context
00275         if (_simfqtServiceContext == NULL) {
00276             throw stdair::NonInitialisedServiceException ("The SimFQT service "
"has not been initialised")
00277         }
00278         assert (_simfqtServiceContext != NULL);
00279
00280         SIMFQT_ServiceContext& lSIMFQT_ServiceContext = *
_simfqtServiceContext;
00281
00282         // Retrieve the STDAIR service object from the (SimFQT) service context
00283         stdair::STDAIR_Service& lSTDAIR_Service =
lSIMFQT_ServiceContext.getSTDAIR_Service();
00284
00285         // Get the root of the BOM tree, on which all of the other BOM objects
00286         // are attached
00287         stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00288
00289         // Delegate the BOM display to the dedicated service
00290         std::ostream oCSVStr;
00291         stdair::BomDisplay::csvSimFQTAirRACDisplay (oCSVStr, lBomRoot);
00292         return oCSVStr.str();
00293     }
00294
00295     // ////////////////////////////////////////
00296     std::string SIMFQT_Service::
00297     csvDisplay (const stdair::TravelSolutionList_T&
ioTravelSolutionList) const {
00298
00299         // Retrieve the Simfqt service context

```

```

00306     if (_simfqtServiceContext == NULL) {
00307         throw stdair::NonInitialisedServiceException ("The Simfqt service has not
"
00308                                                     "been initialised");
00309     }
00310     assert (_simfqtServiceContext != NULL);
00311     // Retrieve the Simfqt service context
00312     SIMFQT_ServiceContext& lSIMFQT_ServiceContext = *
_simfqtServiceContext;
00314
00315     // Retrieve the STDAIR service object from the (Simfqt) service context
00316     stdair::STDAIR_Service& lSTDAIR_Service =
00317         lSIMFQT_ServiceContext.getSTDAIR_Service();
00318
00319     // Delegate the BOM building to the dedicated service
00320     return lSTDAIR_Service.csvDisplay (ioTravelSolutionList);
00321 }
00322
00323 // //////////////////////////////////////
00324 std::string SIMFQT_Service::
00325 csvDisplay (const stdair::AirportCode_T& iOrigin,
00326             const stdair::AirportCode_T& iDestination,
00327             const stdair::Date_T& iDepartureDate) const {
00328
00329     // Retrieve the SIMFQT service context
00330     if (_simfqtServiceContext == NULL) {
00331         throw stdair::NonInitialisedServiceException ("The Simfqt service "
00332                                                     "has not been initialised")
;
00333     }
00334     assert (_simfqtServiceContext != NULL);
00335
00336     SIMFQT_ServiceContext& lSIMFQT_ServiceContext = *
_simfqtServiceContext;
00337
00338     // Retrieve the STDAIR service object from the (SIMFQT) service context
00339     stdair::STDAIR_Service& lSTDAIR_Service =
00340         lSIMFQT_ServiceContext.getSTDAIR_Service();
00341
00342     // Delegate the BOM display to the dedicated service
00343     return lSTDAIR_Service.csvDisplay (iOrigin, iDestination,
00344                                         iDepartureDate);
00345 }
00346
00347 // //////////////////////////////////////
00348 std::string SIMFQT_Service::list() const {
00349
00350     // Retrieve the SIMFQT service context
00351     if (_simfqtServiceContext == NULL) {
00352         throw stdair::NonInitialisedServiceException ("The Simfqt service "
00353                                                     "has not been initialised")
;
00354     }
00355     assert (_simfqtServiceContext != NULL);
00356
00357     SIMFQT_ServiceContext& lSIMFQT_ServiceContext = *
_simfqtServiceContext;
00358
00359     // Retrieve the STDAIR service object from the (SIMFQT) service context
00360     stdair::STDAIR_Service& lSTDAIR_Service =
00361         lSIMFQT_ServiceContext.getSTDAIR_Service();
00362
00363     // Delegate the BOM display to the dedicated service
00364     return lSTDAIR_Service.listAirportPairDateRange ();
00365 }
00366
00367 // //////////////////////////////////////
00368 bool SIMFQT_Service::
00369 check (const stdair::AirportCode_T& iOrigin,
00370        const stdair::AirportCode_T& iDestination,
00371        const stdair::Date_T& iDepartureDate) const {
00372     std::ostringstream oFlightListStr;
00373
00374     if (_simfqtServiceContext == NULL) {
00375         throw stdair::NonInitialisedServiceException ("The Simfqt service "
00376                                                     "has not been initialised")
;
00377     }
00378     assert (_simfqtServiceContext != NULL);
00379     SIMFQT_ServiceContext& lSIMFQT_ServiceContext = *
_simfqtServiceContext;
00380
00381     // Retrieve the STDAIR service object from the (SIMFQT) service context
00382     stdair::STDAIR_Service& lSTDAIR_Service =
00383         lSIMFQT_ServiceContext.getSTDAIR_Service();
00384

```

```

00385     // Delegate the BOM display to the dedicated service
00386     return lSTDAIR_Service.check (iOrigin, iDestination, iDepartureDate);
00387 }
00388
00389 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00390 void SIMFQT_Service::
00391 quotePrices (const stdair::BookingRequestStruct& iBookingRequest
00392             ,
00393             stdair::TravelSolutionList_T& ioTravelSolutionList) {
00394     // Retrieve the Simfqt service context
00395     if (_simfqtServiceContext == NULL) {
00396         throw stdair::NonInitialisedServiceException ("The SimFQT service "
00397             "has not been initialised")
00398     };
00399     assert (_simfqtServiceContext != NULL);
00400
00401     SIMFQT_ServiceContext& lSIMFQT_ServiceContext = *
00402     _simfqtServiceContext;
00403
00404     // Retrieve the StdAir service context
00405     stdair::STDAIR_Service& lSTDAIR_Service =
00406     lSIMFQT_ServiceContext.getSTDAIR_Service();
00407
00408     // Get the root of the BOM tree, on which all of the other BOM objects
00409     // will be attached
00410     stdair::BomRoot& lBomRoot = lSTDAIR_Service.getBomRoot();
00411
00412     // Delegate the action to the dedicated command
00413     stdair::BasChronometer lFareQuoteRetrievalChronometer;
00414     lFareQuoteRetrievalChronometer.start();
00415     FareQuoter::priceQuote (iBookingRequest, ioTravelSolutionList, lBomRoot);
00416
00417     // DEBUG
00418     const double lFareQuoteRetrievalMeasure =
00419     lFareQuoteRetrievalChronometer.elapsed();
00420     STDAIR_LOG_DEBUG ("Fare Quote retrieving: " << lFareQuoteRetrievalMeasure
00421         << " - " << lSIMFQT_ServiceContext.display());
00422 }
00423 }

```

23.51 simfqt/service/SIMFQT_ServiceContext.cpp File Reference

```

#include <cassert>
#include <sstream>
#include <simfqt/basic/BasConst_SIMFQT_Service.hpp>
#include <simfqt/service/SIMFQT_ServiceContext.hpp>

```

Namespaces

- namespace [SIMFQT](#)

23.52 SIMFQT_ServiceContext.cpp

```

00001 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00002 // Import section
00003 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // SimFQT
00008 #include <simfqt/basic/BasConst_SIMFQT_Service.hpp>
00009 #include <simfqt/service/SIMFQT_ServiceContext.hpp>
00010
00011 namespace SIMFQT {
00012
00013     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00014     SIMFQT_ServiceContext::SIMFQT_ServiceContext() : _ownStdairService (false) {
00015     }
00016
00017     //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

```

00018 SIMFQT_ServiceContext::SIMFQT_ServiceContext (const SIMFQT_ServiceContext&) {
00019     assert (false);
00020 }
00021
00022 // //////////////////////////////////////
00023 SIMFQT_ServiceContext::~SIMFQT_ServiceContext() {
00024 }
00025
00026 // //////////////////////////////////////
00027 stdair::STDAIR_Service& SIMFQT_ServiceContext::getSTDAIR_Service() const {
00028     assert (_stdairService != NULL);
00029     return *_stdairService;
00030 }
00031
00032 // //////////////////////////////////////
00033 const std::string SIMFQT_ServiceContext::shortDisplay() const {
00034     std::ostringstream ostr;
00035     ostr << "SIMFQT_ServiceContext -- Owns StdAir service: "
00036         << _ownStdairService;
00037     return ostr.str();
00038 }
00039
00040 // //////////////////////////////////////
00041 const std::string SIMFQT_ServiceContext::display() const {
00042     std::ostringstream ostr;
00043     ostr << shortDisplay();
00044     return ostr.str();
00045 }
00046
00047 // //////////////////////////////////////
00048 const std::string SIMFQT_ServiceContext::describe() const {
00049     return shortDisplay();
00050 }
00051
00052 // //////////////////////////////////////
00053 void SIMFQT_ServiceContext::reset() {
00054     if (_ownStdairService == true) {
00055         _stdairService.reset();
00056     }
00057 }
00058
00059 }

```

23.53 simfqt/service/SIMFQT_ServiceContext.hpp File Reference

```

#include <string>
#include <stdair/stdair_service_types.hpp>
#include <stdair/service/ServiceAbstract.hpp>
#include <simfqt/SIMFQT_Types.hpp>

```

Classes

- class [SIMFQT::SIMFQT_ServiceContext](#)
Class holding the context of the SimFQT services.

Namespaces

- namespace [stdair](#)
Forward declarations.
- namespace [SIMFQT](#)

23.54 SIMFQT_ServiceContext.hpp

```

00001 #ifndef __SIMFQT_SVC_SIMFQTSERVICECONTEXT_HPP
00002 #define __SIMFQT_SVC_SIMFQTSERVICECONTEXT_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 // SimFQT
00013 #include <simfqt/SIMFQT_Types.hpp>
00014
00016 namespace stdair {
00017     class STDAIR_Service;
00018 }
00019
00020 namespace SIMFQT {
00021
00025     class SIMFQT_ServiceContext : public
stdair::ServiceAbstract {
00031     friend class SIMFQT_Service;
00032     friend class FacSimfqtServiceContext;
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
00051         const bool getOwnStdairServiceFlag() const {
00052             return _ownStdairService;
00053         }
00054
00055     private:
00056         // ////////// Setters //////////
00061         void setSTDAIR_Service (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr,
00062                                 const bool iOwnStdairService) {
00063             _stdairService = ioSTDAIR_ServicePtr;
00064             _ownStdairService = iOwnStdairService;
00065         }
00066
00070         void reset();
00071
00072     private:
00073         // ////////// Display Methods //////////
00078         const std::string shortDisplay() const;
00079
00083         const std::string display() const;
00084
00088         const std::string describe() const;
00089
00090     private:
00092         // ////////// Construction / initialisation //////////
00096         SIMFQT_ServiceContext (const FareQuoteID_T&);
00097
00101         SIMFQT_ServiceContext ();
00102
00106         SIMFQT_ServiceContext (const SIMFQT_ServiceContext&);
00107
00111         ~SIMFQT_ServiceContext ();
00112
00113     private:
00114         // ////////// Children //////////
00115         stdair::STDAIR_ServicePtr_T _stdairService;
00119
00120         bool _ownStdairService;
00125     };
00126
00127 }
00128 #endif // __SIMFQT_SVC_SIMFQTSERVICECONTEXT_HPP

```

23.55 simfqt/SIMFQT_Service.hpp File Reference

```

#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <simfqt/SIMFQT_Types.hpp>

```


Classes

- class `SIMFQT::SIMFQT_Service`
Interface for the `SIMFQT` Services.

Namespaces

- namespace `stdair`
Forward declarations.
- namespace `SIMFQT`

23.56 SIMFQT_Service.hpp

```

00001 #ifndef __SIMFQT_SVC_SIMFQT_SERVICE_HPP
00002 #define __SIMFQT_SVC_SIMFQT_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 // SimFQT
00012 #include <simfqt/SIMFQT_Types.hpp>
00013
00015 namespace stdair {
00016     class STDAIR_Service;
00017     struct BookingRequestStruct;
00018     struct BasLogParams;
00019     struct BasDBParams;
00020 }
00021
00022 namespace SIMFQT {
00023
00025     class SIMFQT_ServiceContext;
00026
00027     class SIMFQT_Service {
00031     public:
00032
00033         // ////////////////////////////////// Constructors and Destructors //////////////////////////////////
00034         SIMFQT_Service (const stdair::BasLogParams&);
00046
00047         SIMFQT_Service (const stdair::BasLogParams&, const
00060         stdair::BasDBParams&);
00061
00077         SIMFQT_Service (stdair::STDAIR_ServicePtr_T
00078         ioSTDAIR_ServicePtr);
00079
00087         void parseAndLoad (const FareFilePath&
00088         iFareFilename);
00092
00093         ~SIMFQT_Service();
00094
00095     public:
00096         // ////////////////////////////////// Business Methods //////////////////////////////////
00108         void buildSampleBom();
00109
00116         stdair::BookingRequestStruct buildBookingRequest (const
00117         bool isForCRS = false);
00135
00136         void buildSampleTravelSolutions (
00146         stdair::TravelSolutionList_T&);
00147
00148         void quotePrices (const stdair::BookingRequestStruct&,
00149         stdair::TravelSolutionList_T&);
00150
00151     public:
00151         // ////////////////////////////////// Display support methods //////////////////////////////////
00159         std::string csvDisplay() const;
00160
00168         std::string csvDisplay (const stdair::TravelSolutionList_T&
00169         const);
00182         std::string csvDisplay (const stdair::AirportCode_T& ioOrigin,

```

```

00183                                     const stdair::AirportCode_T& ioDestination,
00184                                     const stdair::Date_T& ioDepartureDate) const;
00185
00194     std::string list() const;
00195
00208     bool check (const stdair::AirportCode_T& ioOrigin,
00209                 const stdair::AirportCode_T& ioDestination,
00210                 const stdair::Date_T& ioDepartureDate) const;
00211
00212 private:
00213     // ////////// Construction and Destruction helper methods //////////
00217     SIMFQT_Service();
00218
00222     SIMFQT_Service (const SIMFQT_Service&);
00223
00233     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&,
00234                                                    const stdair::BasDBParams&);
00235
00244     stdair::STDAIR_ServicePtr_T initStdAirService (const stdair::BasLogParams&)
00245 ;
00254     void addStdAirService (stdair::STDAIR_ServicePtr_T ioSTDAIR_ServicePtr,
00255                           const bool iOwnStdairService);
00256
00261     void initServiceContext();
00262
00269     void initSimfqtService();
00270
00279     void initSimfqtService (const FareFilePath& iFareFilename);
00280
00284     void finalise();
00285
00286
00287 private:
00288     // ////////// Service Context //////////
00292     SIMFQT_ServiceContext* _simfqtServiceContext;
00293 };
00294 }
00295 #endif // __SIMFQT_SVC_SIMFQT_SERVICE_HPP

```

23.57 simfqt/SIMFQT_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 [SIMFQT::FareFileParsingFailedException](#)
- class [SIMFQT::AirportPairNotFoundException](#)
- class [SIMFQT::PosOrChannelNotFoundException](#)
- class [SIMFQT::FlightDateNotFoundException](#)
- class [SIMFQT::FlightTimeNotFoundException](#)
- class [SIMFQT::FeaturesNotFoundException](#)
- class [SIMFQT::AirlineNotFoundException](#)
- class [SIMFQT::FareInputFileNotFoundException](#)
- class [SIMFQT::QuotingException](#)
- class [SIMFQT::FareFilePath](#)

Namespaces

- namespace [SIMFQT](#)

Typedefs

- typedef unsigned int [SIMFQT::FareQuotID_T](#)
- typedef boost::shared_ptr
 < SIMFQT_Service > [SIMFQT::SIMFQT_ServicePtr_T](#)

23.58 SIMFQT_Types.hpp

```

00001 #ifndef __SIMFQT_SIMFQT_TYPES_HPP
00002 #define __SIMFQT_SIMFQT_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 SIMFQT {
00017
00018     // Forward declarations
00019     class SIMFQT_Service;
00020
00021
00022     // ////////// Exceptions //////////
00026     class FareFileParsingFailedException
00027     : public stdair::ParsingFileFailedException {
00028     public:
00032     FareFileParsingFailedException (const
std::string& iWhat)
00033     : stdair::ParsingFileFailedException (iWhat) {}
00034     };
00035
00039     class AirportPairNotFoundException : public
stdair::ObjectNotFoundExceptio
00040     public:
00044     AirportPairNotFoundExceptio (const std::string
& iWhat)
00045     : stdair::ObjectNotFoundExceptio (iWhat) {}
00046     };
00047
00051     class PosOrChannelNotFoundExceptio : public
stdair::ObjectNotFoundExceptio {
00052     public:
00056     PosOrChannelNotFoundExceptio (const
std::string& iWhat)
00057     : stdair::ObjectNotFoundExceptio (iWhat) {}
00058     };
00059
00063     class FlightDateNotFoundExceptio : public
stdair::ObjectNotFoundExceptio {
00064     public:
00068     FlightDateNotFoundExceptio (const std::string&
iWhat)
00069     : stdair::ObjectNotFoundExceptio (iWhat) {}
00070     };
00071
00075     class FlightTimeNotFoundExceptio : public
stdair::ObjectNotFoundExceptio {
00076     public:
00080     FlightTimeNotFoundExceptio (const std::string&
iWhat)
00081     : stdair::ObjectNotFoundExceptio (iWhat) {}
00082     };
00083
00087     class FeaturesNotFoundExceptio : public
stdair::ObjectNotFoundExceptio {
00088     public:
00092     FeaturesNotFoundExceptio (const std::string&
iWhat)
00093     : stdair::ObjectNotFoundExceptio (iWhat) {}
00094     };
00095
00099     class AirlineNotFoundExceptio : public
stdair::ObjectNotFoundExceptio {
00100     public:
00104     AirlineNotFoundExceptio (const std::string& iWhat)

```

```

00105         : stdair::ObjectNotFoundException (iWhat) {}
00106     };
00107
00111     class FareInputFileNotFoundException : public
stdair::FileNotFoundException {
00112     public:
00116         FareInputFileNotFoundException (const
std::string& iWhat)
00117         : stdair::FileNotFoundException (iWhat) {}
00118     };
00119
00123     class QuotingException : public stdair::RootException {
00124     };
00125
00126     // ////////// Files //////////
00130     class FareFilePath : public stdair::InputFilePath {
00131     public:
00135         explicit FareFilePath (const stdair::Filename_T& iFilename)
00136         : stdair::InputFilePath (iFilename) {}
00137     };
00138
00139     // ////////// Type definitions specific to SimFQT //////////
00143     typedef unsigned int FareQuoteID_T;
00144
00148     typedef boost::shared_ptr<SIMFQT_Service> SIMFQT_ServicePtr_T
;
00149 }
00150 #endif // __SIMFQT_SIMFQT_TYPES_HPP

```

23.59 simfqt/ui/cmdline/simfqt.cpp File Reference

23.60 simfqt.cpp

```

00001
00005 // STL
00006 #include <cassert>
00007 #include <iostream>
00008 #include <sstream>
00009 #include <fstream>
00010 #include <string>
00011 // Boost (Extended STL)
00012 #include <boost/program_options.hpp>
00013 #include <boost/tokenizer.hpp>
00014 #include <boost/regex.hpp>
00015 // StdAir
00016 #include <stdair/basic/BasLogParams.hpp>
00017 #include <stdair/basic/BasConst_BomDisplay.hpp>
00018 #include <stdair/basic/BasDBParams.hpp>
00019 #include <stdair/basic/BasConst_DefaultObject.hpp>
00020 #include <stdair/basic/BasConst_Inventory.hpp>
00021 #include <stdair/basic/BasConst_Request.hpp>
00022 #include <stdair/service/Logger.hpp>
00023 #include <stdair/stdair_exceptions.hpp>
00024 #include <stdair/stdair_basic_types.hpp>
00025 #include <stdair/stdair_date_time_types.hpp>
00026 #include <stdair/bom/TravelSolutionStruct.hpp>
00027 #include <stdair/bom/BookingRequestStruct.hpp>
00028 #include <stdair/bom/ParsedKey.hpp>
00029 #include <stdair/bom/BomKeyManager.hpp>
00030 #include <stdair/command/CmdBomManager.hpp>
00031 // Stdair GNU Readline Wrapper
00032 #include <stdair/ui/cmdline/SReadline.hpp>
00033 // Simfqt
00034 #include <simfqt/SIMFQT_Service.hpp>
00035 #include <simfqt/config/simfqt-paths.hpp>
00036
00037
00038 // ////////// Constants //////////
00042 const std::string K_SIMFQT_DEFAULT_LOG_FILENAME ("
simfqt.log");
00043
00047 const std::string K_SIMFQT_DEFAULT_FARE_INPUT_FILENAME
(STDPAIR_SAMPLE_DIR
00048                                     "/fare01.csv");
00049
00054 const bool K_SIMFQT_DEFAULT_BUILT_IN_INPUT =
false;
00055
00059 const int K_SIMFQT_EARLY_RETURN_STATUS = 99;
00060
00065 typedef std::vector<std::string> TokenList_T;
00066
00070 struct Command_T {

```

```

00071     typedef enum {
00072         NOP = 0,
00073         QUIT,
00074         HELP,
00075         LIST,
00076         DISPLAY,
00077         PRICE,
00078         LAST_VALUE
00079     } Type_T;
00080 };
00081
00082 // ////////// Parsing of Options & Configuration //////////
00083 // A helper function to simplify the main part.
00084 template<class T> std::ostream& operator<< (std::ostream& os,
00085     const std::vector<T>& v) {
00086     std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00087     return os;
00088 }
00089
00093 int readConfiguration (int argc, char* argv[], bool&
    ioIsBuiltin,
00094     stdair::Filename_T& ioFareInputFilename,
00095     std::string& ioLogFilename) {
00096
00097     // Default for the built-in input
00098     ioIsBuiltin = K_SIMFQT_DEFAULT_BUILT_IN_INPUT;
00099
00100     // Declare a group of options that will be allowed only on command line
00101     boost::program_options::options_description generic ("Generic options");
00102     generic.add_options()
00103         ("prefix", "print installation prefix")
00104         ("version,v", "print version string")
00105         ("help,h", "produce help message");
00106
00107     // Declare a group of options that will be allowed both on command
00108     // line and in config file
00109     boost::program_options::options_description config ("Configuration");
00110     config.add_options()
00111         ("builtin,b",
00112         "The sample BOM tree can be either built-in or parsed from an input file.
00113         That latter must then be given with the -f/--fare option")
00114         ("fare,f",
00115         boost::program_options::value< std::string >(&ioFareInputFilename)->
00116         default_value(K_SIMFQT_DEFAULT_FARE_INPUT_FILENAME
00117         ),
00118         "(CSV) input file for the fare rules")
00119         ("log,l",
00120         boost::program_options::value< std::string >(&ioLogFilename)->
00121         default_value(K_SIMFQT_DEFAULT_LOG_FILENAME),
00122         "Filename for the logs")
00123         ;
00124
00125     // Hidden options, will be allowed both on command line and
00126     // in config file, but will not be shown to the user.
00127     boost::program_options::options_description hidden ("Hidden options");
00128     hidden.add_options()
00129         ("copyright",
00130         boost::program_options::value< std::vector<std::string> >(),
00131         "Show the copyright (license)");
00132
00133     boost::program_options::options_description cmdline_options;
00134     cmdline_options.add(generic).add(config).add(hidden);
00135
00136     boost::program_options::options_description config_file_options;
00137     config_file_options.add(config).add(hidden);
00138
00139     boost::program_options::options_description visible ("Allowed options");
00140     visible.add(generic).add(config);
00141
00142     boost::program_options::positional_options_description p;
00143     p.add ("copyright", -1);
00144
00145     boost::program_options::variables_map vm;
00146     boost::program_options::store (boost::program_options::command_line_parser (argc, argv).
00147         options (cmdline_options).positional(p).run(), vm);
00148
00149     std::ifstream ifs ("simfqt.cfg");
00150     boost::program_options::store (parse_config_file (ifs, config_file_options),
00151         vm);
00152     boost::program_options::notify (vm); if (vm.count ("help")) {
00153         std::cout << visible << std::endl;
00154         return K_SIMFQT_EARLY_RETURN_STATUS;
00155     }
00156
00157     if (vm.count ("version")) {
00158         std::cout << PACKAGE_NAME << " , version " << PACKAGE_VERSION

```

```

00156     << std::endl;
00157     return K_SIMFQT_EARLY_RETURN_STATUS;
00158 }
00159 if (vm.count ("prefix")) {
00160     std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
00161     return K_SIMFQT_EARLY_RETURN_STATUS;
00162 }
00163
00164 if (vm.count ("builtin")) {
00165     ioIsBuiltin = true;
00166 }
00167 const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
00168 std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;
00169
00170 if (ioIsBuiltin == false) {
00171
00172     // The BOM tree should be built from parsing a fare (and O&D) file
00173     if (vm.count ("fare")) {
00174         ioFareInputFilename = vm["fare"].as< std::string >();
00175         std::cout << "Input fare filename is: " << ioFareInputFilename
00176             << std::endl;
00177     } else {
00178         // The built-in option is not selected. However, no fare file
00179         // is specified
00180         std::cerr << "Either one among the -b/--builtin and -f/--fare "
00181             << "options must be specified" << std::endl;
00182     }
00183 }
00184
00185 if (vm.count ("log")) {
00186     ioLogFilename = vm["log"].as< std::string >();
00187     std::cout << "Log filename is: " << ioLogFilename << std::endl;
00188 }
00189
00190 return 0;
00191 }
00192
00193 // //////////////////////////////////////
00194 void initReadline (swift::SReadline& ioInputReader) {
00195
00196     // Prepare the list of my own completers
00197     std::vector<std::string> Completers;
00198
00199     // The following is supported:
00200     // - "identifiers"
00201     // - special identifier %file - means to perform a file name completion
00202     Completers.push_back ("help");
00203     Completers.push_back ("list");
00204     Completers.push_back ("display %airport_code %airport_code %departure_date");
00205     Completers.push_back ("price %airline_code %flight_number %departure_date
00206         %airport_code %airport_code %departure_time %booking_date %booking_time %POS
00207         %channel% %trip_type %stay_duration");
00208     Completers.push_back ("quit");
00209
00210     // Now register the completers.
00211     // Actually it is possible to re-register another set at any time
00212     ioInputReader.RegisterCompletions (Completers);
00213 }
00214
00215 // //////////////////////////////////////
00216 Command_T::Type_T extractCommand (TokenList_T& ioTokenList) {
00217     Command_T::Type_T oCommandType = Command_T::LAST_VALUE;
00218
00219     // Interpret the user input
00220     if (ioTokenList.empty() == false) {
00221         TokenList_T::iterator itTok = ioTokenList.begin();
00222         std::string& lCommand (*itTok);
00223         boost::algorithm::to_lower (lCommand);
00224
00225         if (lCommand == "help") {
00226             oCommandType = Command_T::HELP;
00227         } else if (lCommand == "list") {
00228             oCommandType = Command_T::LIST;
00229         } else if (lCommand == "display") {
00230             oCommandType = Command_T::DISPLAY;
00231         } else if (lCommand == "price") {
00232             oCommandType = Command_T::PRICE;
00233         } else if (lCommand == "quit") {
00234             oCommandType = Command_T::QUIT;
00235         }
00236     }
00237 }
00238
00239

```

```

00240     }
00241
00242     // Remove the first token (the command), as the corresponding information
00243     // has been extracted in the form of the returned command type enumeration
00244     ioTokenList.erase (itTok);
00245
00246     } else {
00247         oCommandType = Command_T::NOP;
00248     }
00249
00250     return oCommandType;
00251 }
00252
00253 //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
00254 // Re-compose a date using three strings: the year, the month and the
00255 // day. Return true if a correct date has been computed, false if not.
00256 bool retrieveDate (std::string iYearString,
00257                  std::string iMonthString,
00258                  std::string iDayString,
00259                  stdair::Date_T& ioDate) {
00260
00261     const std::string kMonthStr[12] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
00262                                       "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};
00263
00264     // Check the year.
00265     unsigned short lDateYear;
00266     try {
00267         lDateYear = boost::lexical_cast<unsigned short> (iYearString);
00268         if (lDateYear < 100) {
00269             lDateYear += 2000;
00270         }
00271     }
00272     } catch (boost::bad_lexical_cast& eCast) {
00273         std::cerr << "The year ('" << iYearString
00274                 << "') cannot be understood." << std::endl;
00275         return false;
00276     }
00277
00278     // Check the month.
00279     std::string lDateMonthStr;
00280     try {
00281         const boost::regex lMonthRegex ("^(\\d{1,2})$");
00282         const bool isMonthANumber = regex_match (iMonthString, lMonthRegex);
00283
00284         if (isMonthANumber == true) {
00285             const unsigned short lMonth =
00286                 boost::lexical_cast<unsigned short> (iMonthString);
00287             if (lMonth > 12) {
00288                 throw boost::bad_lexical_cast();
00289             }
00290             if (lMonth != 0) {
00291                 lDateMonthStr = kMonthStr[lMonth-1];
00292             } else {
00293                 std::cerr << "The month ('" << iMonthString
00294                         << "') cannot be understood." << std::endl;
00295                 return false;
00296             }
00297         } else {
00298             if (iMonthString.size() < 3) {
00299                 throw boost::bad_lexical_cast();
00300             }
00301             std::string lMonthStr1 (iMonthString.substr (0, 1));
00302             boost::algorithm::to_upper (lMonthStr1);
00303             std::string lMonthStr23 (iMonthString.substr (1, 2));
00304             boost::algorithm::to_lower (lMonthStr23);
00305             lDateMonthStr = lMonthStr1 + lMonthStr23;
00306         }
00307     } catch (boost::bad_lexical_cast& eCast) {
00308         std::cerr << "The month ('" << iMonthString
00309                 << "') cannot be understood." << std::endl;
00310         return false;
00311     }
00312
00313     // Check the day.
00314     unsigned short lDateDay;
00315     try {
00316         lDateDay = boost::lexical_cast<unsigned short> (iDayString);
00317     } catch (boost::bad_lexical_cast& eCast) {
00318         std::cerr << "The day ('" << iDayString
00319                 << "') cannot be understood." << std::endl;
00320         return false;
00321     }

```

```

00327     }
00328
00329     // Re-compose the date.
00330     std::ostringstream lDateStr;
00331     lDateStr << lDateYear << "-" << lDateMonthStr
00332         << "-" << lDateDay;
00333     try {
00334
00335         ioDate =
00336             boost::gregorian::from_simple_string (lDateStr.str());
00337
00338     } catch (boost::gregorian::bad_month& eCast) {
00339         std::cerr << "The month of the date ('" << lDateStr.str()
00340             << "') cannot be understood." << std::endl;
00341         return false;
00342     } catch (boost::gregorian::bad_day_of_month& eCast) {
00343         std::cerr << "The date ('" << lDateStr.str()
00344             << "') is not correct: the day of month does not exist."
00345             << std::endl;
00346         return false;
00347     } catch (boost::gregorian::bad_year& eCast) {
00348         std::cerr << "The year ('" << lDateStr.str()
00349             << "') is not correct."
00350             << std::endl;
00351         return false;
00352     }
00353
00354     return true;
00355 }
00356
00357 // //////////////////////////////////////
00358 // Re-compose a time using two strings: the hour and the minute.
00359 // Return true if a correct time has been computed, false if not.
00360 bool retrieveTime (std::string iHourString,
00361                 std::string iMinuteString,
00362                 stdair::Duration_T& oTime) {
00363
00364     // Check the hour
00365     unsigned short lTimeHour;
00366     try {
00367
00368         lTimeHour = boost::lexical_cast<unsigned short> (iHourString);
00369
00370     } catch (boost::bad_lexical_cast& eCast) {
00371         std::cerr << "The hour of the time ('" << iHourString
00372             << "') cannot be understood." << std::endl;
00373         return false;
00374     }
00375
00376     // Check the minutes
00377     unsigned short lTimeMinute;
00378     try {
00379
00380         lTimeMinute = boost::lexical_cast<unsigned short> (iMinuteString);
00381
00382     } catch (boost::bad_lexical_cast& eCast) {
00383         std::cerr << "The minute of the time ('" << iMinuteString
00384             << "') cannot be understood." << std::endl;
00385         return false;
00386     }
00387
00388     // Re-compose the time
00389     std::ostringstream lTimeStr;
00390     lTimeStr << lTimeHour << ":" << lTimeMinute;
00391     oTime =
00392         boost::posix_time::duration_from_string (lTimeStr.str());
00393
00394     return true;
00395 }
00396
00397 // //////////////////////////////////////
00398 // Analyze the tokens of the 'price' command in order to construct
00399 // a travel solution list and a booking request.
00400 const stdair::BookingRequestStruct parseTravelSolutionAndBookingRequestKey
00401 (const TokenList_T& iTokenList,
00402  stdair::TravelSolutionList_T& ioInteractiveTravelSolutionList,
00403  const stdair::BookingRequestStruct& ioBookingRequestStruct) {
00404
00405     TokenList_T::const_iterator iTok = iTokenList.begin();
00406
00407     if (iTok->empty() == true) {
00408
00409         std::cerr << "Wrong list of parameters. "
00410             << "The default booking request and travel solution list are
00411 kept."
00412             << std::endl;

```



```

00413     return ioBookingRequestStruct;
00414
00415
00416 } else {
00417     // Parameters corresponding to the tokens.
00418     // Each parameter corresponds to one token except the dates
00419     // (three tokens) and the times (two tokens).
00420     stdair::AirlineCode_T lAirlineCode;
00421     stdair::FlightNumber_T lflightNumber;
00422     stdair::Date_T lDepartureDate;
00423     stdair::Duration_T lDepartureTime;
00424     stdair::AirportCode_T lOriginAirport;
00425     stdair::AirportCode_T lDestinationAirport;
00426     stdair::Date_T lRequestDate;
00427     stdair::Duration_T lRequestTime;
00428     stdair::CityCode_T lPOS;
00429     stdair::ChannelLabel_T lChannel;
00430     stdair::TripType_T lTripType;
00431     unsigned short lStayDuration;
00432
00433     // Read the airline code.
00434     lAirlineCode = *itTok;
00435     boost::algorithm::to_upper (lAirlineCode);
00436
00437     // Read the flight-number .
00438     ++itTok;
00439     if (itTok->empty() == false) {
00440         try {
00441             lflightNumber = boost::lexical_cast<stdair::FlightNumber_T> (*itTok);
00442
00443         } catch (boost::bad_lexical_cast& eCast) {
00444             std::cerr << "The flight number ('" << *itTok
00445                 << "') cannot be understood."
00446                 << std::endl;
00447             return ioBookingRequestStruct;
00448         }
00449     }
00450
00451     // Read the departure date.
00452     ++itTok;
00453     if (itTok->empty() == true) {
00454         return ioBookingRequestStruct;
00455     }
00456     const std::string lDepartureYearString = *itTok;
00457     ++itTok;
00458     if (itTok->empty() == true) {
00459         return ioBookingRequestStruct;
00460     }
00461     const std::string lDepartureMonthString = *itTok;
00462     ++itTok;
00463     if (itTok->empty() == true) {
00464         return ioBookingRequestStruct;
00465     }
00466     const std::string lDepartureDayString = *itTok;
00467     const bool IsDepartureDateReadable =
00468         retrieveDate (lDepartureYearString, lDepartureMonthString,
00469             lDepartureDayString, lDepartureDate);
00470
00471     if (IsDepartureDateReadable == false) {
00472         std::cerr << "The default booking request and travel solution list are
00473 kept."
00474             << std::endl;
00475         return ioBookingRequestStruct;
00476     }
00477
00478     // Read the origin.
00479     ++itTok;
00480     if (itTok->empty() == false) {
00481         lOriginAirport = *itTok;
00482         boost::algorithm::to_upper (lOriginAirport);
00483     }
00484
00485     // Read the destination.
00486     ++itTok;
00487     if (itTok->empty() == false) {
00488         lDestinationAirport = *itTok;
00489         boost::algorithm::to_upper (lDestinationAirport);
00490     }
00491
00492     // Read the departure time.
00493     ++itTok;
00494     if (itTok->empty() == true) {
00495         return ioBookingRequestStruct;
00496     }
00497     const std::string lDepartureHourString = *itTok;
00498     ++itTok;

```

```

00499     if (itTok->empty() == true) {
00500         return ioBookingRequestStruct;
00501     }
00502     const std::string lDepartureMinuteString = *itTok;
00503     const bool IsDepartureTimeReadable =
00504         retrieveTime (lDepartureHourString, lDepartureMinuteString,
00505                     lDepartureTimeString);
00506
00507     if (IsDepartureTimeReadable == false) {
00508         std::cerr << "The default booking request and travel solution list are
kept."
00509             << std::endl;
00510         return ioBookingRequestStruct;
00511     }
00512
00513     // Read the request date.
00514     ++itTok;
00515     if (itTok->empty() == true) {
00516         return ioBookingRequestStruct;
00517     }
00518     const std::string lRequestYearString = *itTok;
00519     ++itTok;
00520     if (itTok->empty() == true) {
00521         return ioBookingRequestStruct;
00522     }
00523     const std::string lRequestMonthString = *itTok;
00524     ++itTok;
00525     if (itTok->empty() == true) {
00526         return ioBookingRequestStruct;
00527     }
00528     const std::string lRequestDayString = *itTok;
00529     const bool IsRequestDateReadable =
00530         retrieveDate (lRequestYearString, lRequestMonthString,
00531                     lRequestDayString, lRequestDate);
00532
00533     if (IsRequestDateReadable == false) {
00534         std::cerr << "The default booking request and travel solution list are
kept."
00535             << std::endl;
00536         return ioBookingRequestStruct;
00537     }
00538
00539     // Read the request time.
00540     ++itTok;
00541     if (itTok->empty() == true) {
00542         return ioBookingRequestStruct;
00543     }
00544     const std::string lRequestHourString = *itTok;
00545     ++itTok;
00546     if (itTok->empty() == true) {
00547         return ioBookingRequestStruct;
00548     }
00549     const std::string lRequestMinuteString = *itTok;
00550     const bool IsRequestTimeReadable =
00551         retrieveTime (lRequestHourString, lRequestMinuteString,
00552                     lRequestTimeString);
00553
00554     if (IsRequestTimeReadable == false) {
00555         std::cerr << "The default booking request and travel solution list are
kept."
00556             << std::endl;
00557         return ioBookingRequestStruct;
00558     }
00559
00560     // Read the POS.
00561     ++itTok;
00562     if (itTok->empty() == false) {
00563         lPOS = *itTok;
00564         boost::algorithm::to_upper (lPOS);
00565     }
00566
00567     // Read the channel.
00568     ++itTok;
00569     if (itTok->empty() == false) {
00570         lChannel = *itTok;
00571         boost::algorithm::to_upper (lChannel);
00572     }
00573
00574     // Read the trip type.
00575     ++itTok;
00576     if (itTok->empty() == false) {
00577         lTripType = *itTok;
00578         boost::algorithm::to_upper (lTripType);
00579     }
00580
00581     // Read the stay duration.
00582     ++itTok;

```

```

00583     if (itTok->empty() == false) {
00584         try {
00585
00586             lStayDuration = boost::lexical_cast<unsigned short> (*itTok);
00587
00588         } catch (boost::bad_lexical_cast& eCast) {
00589             std::cerr << "The stay duration ('" << *itTok
00590                 << "') cannot be understood." << std::endl;
00591             return ioBookingRequestStruct;
00592         }
00593     }
00594
00595     // At this step we know that all the parameters designed to construct
00596     // the travel solution and the booking request are correct.
00597
00598     // Empty the travel solution list to store a new travel solution.
00599     ioInteractiveTravelSolutionList.pop_front();
00600     // Construct the new travel solution.
00601     stdair::TravelSolutionStruct lTravelSolution;
00602     std::ostringstream oStr;
00603     oStr << lAirlineCode
00604         << stdair::DEFAULT_KEY_FLD_DELIMITER
00605         << lflightNumber
00606         << stdair::DEFAULT_KEY_SUB_FLD_DELIMITER
00607         << lDepartureDate
00608         << stdair::DEFAULT_KEY_FLD_DELIMITER
00609         << lOriginAirport
00610         << stdair::DEFAULT_KEY_SUB_FLD_DELIMITER
00611         << lDestinationAirport
00612         << stdair::DEFAULT_KEY_FLD_DELIMITER
00613         << lDepartureTime;
00614     lTravelSolution.addSegment (oStr.str());
00615     ioInteractiveTravelSolutionList.push_front(lTravelSolution);
00616
00617     // Construct the new booking request.
00618     stdair::DateTime_T lRequestDateTime (lRequestDate, lRequestTime);
00619     const stdair::BookingRequestStruct &lBookingRequestStruct =
00620         stdair::BookingRequestStruct (lOriginAirport,
00621             lDestinationAirport,
00622             lPOS,
00623             lDepartureDate,
00624             lRequestDateTime,
00625             stdair::CABIN_ECO,
00626             stdair::DEFAULT_PARTY_SIZE,
00627             lChannel,
00628             lTripType,
00629             lStayDuration,
00630             stdair::FREQUENT_FLYER_MEMBER,
00631             lDepartureTime,
00632             stdair::DEFAULT_WTP,
00633             stdair::DEFAULT_VALUE_OF_TIME);
00634
00635     return lBookingRequestStruct;
00636 }
00637 }
00638
00639 // //////////////////////////////////////
00640 // Analyze the tokens of the 'display' command in order to retrieve
00641 // an airport pair and a departure date.
00642 void parseFlightDateKey (const TokenList_T& iTokenList,
00643     stdair::AirportCode_T& ioOrigin,
00644     stdair::AirportCode_T& ioDestination,
00645     stdair::Date_T& ioDepartureDate) {
00646
00647     TokenList_T::const_iterator itTok = iTokenList.begin();
00648
00649     // Interpret the user input.
00650     if (itTok->empty() == true) {
00651
00652         std::cerr << "Wrong parameters specified. Default paramaters '"
00653             << ioOrigin << "-" << ioDestination
00654             << "/" << ioDepartureDate
00655             << "' are kept."
00656             << std::endl;
00657     } else {
00658
00659         // Read the origin.
00660         ioOrigin = *itTok;
00661         boost::algorithm::to_upper (ioOrigin);
00662
00663         // Read the destination.
00664         ++itTok;
00665         if (itTok->empty() == false) {
00666             ioDestination = *itTok;
00667             boost::algorithm::to_upper (ioDestination);
00668         }
00669     }

```

```

00670
00671 // Read the departure date.
00672 ++itTok;
00673 if (itTok->empty() == true) {
00674     return;
00675 }
00676 std::string lYearString = *itTok;
00677 ++itTok;
00678 if (itTok->empty() == true) {
00679     return;
00680 }
00681 std::string lMonthString = *itTok;
00682 ++itTok;
00683 if (itTok->empty() == true) {
00684     return;
00685 }
00686 std::string lDayString = *itTok;
00687 const bool IsDepartureDateReadable =
00688     retrieveDate (lYearString, lMonthString, lDayString,
00689                 ioDepartureDate);
00690 if (IsDepartureDateReadable == false) {
00691     std::cerr << "Default paramaters '"
00692               << ioOrigin << "-" << ioDestination
00693               << "/" << ioDepartureDate
00694               << "' are kept."
00695               << std::endl;
00696     return;
00697 }
00698 }
00699 }
00700
00701 // //////////////////////////////////////
00702 std::string toString (const TokenList_T& iTokenList) {
00703     std::ostringstream oStr;
00704
00705     // Re-create the string with all the tokens, trimmed by read-line
00706     unsigned short idx = 0;
00707     for (TokenList_T::const_iterator itTok = iTokenList.begin();
00708          itTok != iTokenList.end(); ++itTok, ++idx) {
00709         if (idx != 0) {
00710             oStr << " ";
00711         }
00712         oStr << *itTok;
00713     }
00714
00715     return oStr.str();
00716 }
00717
00718 // //////////////////////////////////////
00719 TokenList_T extractTokenList (const TokenList_T& iTokenList,
00720                             const std::string& iRegularExpression) {
00721     TokenList_T oTokenList;
00722
00723     // Re-create the string with all the tokens (which had been trimmed
00724     // by read-line)
00725     const std::string lFullLine = toString (iTokenList);
00726
00727     // See the caller for the regular expression
00728     boost::regex expression (iRegularExpression);
00729
00730     std::string::const_iterator start = lFullLine.begin();
00731     std::string::const_iterator end = lFullLine.end();
00732
00733     boost::match_results<std::string::const_iterator> what;
00734     boost::match_flag_type flags = boost::match_default | boost::format_sed;
00735     regex_search (start, end, what, expression, flags);
00736
00737     // Put the matched strings in the list of tokens to be returned back
00738     // to the caller
00739     const unsigned short lMatchSetSize = what.size();
00740     for (unsigned short matchIdx = 1; matchIdx != lMatchSetSize; ++matchIdx) {
00741         const std::string lMatchedString (std::string (what[matchIdx].first,
00742                                                       what[matchIdx].second));
00743         //if (lMatchedString.empty() == false) {
00744             oTokenList.push_back (lMatchedString);
00745         //}
00746     }
00747
00748     // DEBUG
00749     std::cout << "After (token list): " << oTokenList << std::endl;
00750
00751     return oTokenList;
00752 }
00753
00754 // //////////////////////////////////////
00755 // Parse the token list of the 'price' command.
00756 TokenList_T extractTokenListForTSAndBR (const TokenList_T& iTokenList) {

```

```

00778     const std::string lRegex("^([[:alpha:]]{2,3})"
00779                               "[[:space:]]+([[:digit:]]{1,4})"
00780                               "[/ ]*"
00781                               "[[:space:]]+([[:digit:]]{2,4})[/-]?"
00782                               "[[:space:]]*([[:alpha:]]{3}|[[:digit:]]{1,2})[/-]?"
00783                               "[[:space:]]*([[:digit:]]{1,2})[[:space:]]*"
00784                               "[[:space:]]+([[:alpha:]]{3})"
00785                               "[[:space:]]+([[:alpha:]]{3})"
00786                               "[[:space:]]+([[:digit:]]{1,2})[[:digit:]]{1,2})[[:digit:]]{1,2})"
00787                               "[[:space:]]+([[:digit:]]{2,4})[/-]?"
00788                               "[[:space:]]*([[:alpha:]]{3}|[[:digit:]]{1,2})[/-]?"
00789                               "[[:space:]]*([[:digit:]]{1,2})"
00790                               "[[:space:]]+([[:digit:]]{1,2})[[:digit:]]{1,2})"
00791                               "[[:space:]]+([[:alpha:]]{3})"
00792                               "[[:space:]]+([[:alpha:]]{2})"
00793                               "[[:space:]]+([[:alpha:]]{2})"
00794                               "[[:space:]]+([[:digit:]]{1})$");
00795
00796     //
00797     const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegex);
00798     return oTokenList;
00799 }
00800
00801 ///////////////////////////////////////////////////////////////////
00802 // Parse the token list of the 'display' command.
00803 TokenList_T extractTokenListForOriDestDate (const TokenList_T& iTokenList) {
00813     const std::string lRegex("^([[:alpha:]]{3})"
00814                               "[[:space:]]*[/-]?"
00815                               "[[:space:]]*([[:alpha:]]{3})"
00816                               "[[:space:]]*[/-]?"
00817                               "[[:space:]]*([[:digit:]]{2,4})"
00818                               "[[:space:]]*[/-]?"
00819                               "[[:space:]]*([[:alpha:]]{3}|[[:digit:]]{1,2})"
00820                               "[[:space:]]*[/-]?"
00821                               "[[:space:]]*([[:digit:]]{1,2})$");
00822
00823     //
00824     const TokenList_T& oTokenList = extractTokenList (iTokenList, lRegex);
00825     return oTokenList;
00826 }
00827
00828 // ////////// M A I N //////////
00829 int main (int argc, char* argv[]) {
00830
00831     // State whether the BOM tree should be built-in or parsed from an
00832     // input file
00833     bool isBuiltin;
00834
00835     // Fare input file name
00836     stdair::Filename_T lFareInputFilename;
00837
00838     // Readline history
00839     const unsigned int lHistorySize (100);
00840     const std::string lHistoryFilename ("simfqt.hist");
00841     const std::string lHistoryBackupFilename ("simfqt.hist.bak");
00842
00843     // Default parameters for the interactive session
00844     stdair::AirportCode_T lInteractiveOrigin;
00845     stdair::AirportCode_T lInteractiveDestination;
00846     stdair::Date_T lInteractiveDepartureDate;
00847
00848     // Output log File
00849     stdair::Filename_T lLogFilename;
00850
00851     // Call the command-line option parser
00852     const int lOptionParserStatus =
00853         readConfiguration (argc, argv, isBuiltin,
00854                             lFareInputFilename, lLogFilename);
00855     if (lOptionParserStatus == K_SIMFQT_EARLY_RETURN_STATUS
00856     ) {
00856         return 0;
00857     }
00858
00859     // Set the log parameters
00860     std::ofstream logOutputFile;
00861     // Open and clean the log outputfile
00862     logOutputFile.open (lLogFilename.c_str());
00863     logOutputFile.clear();
00864
00865     // Initialise the fareQuote service
00866     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00867     SIMFQT::SIMFQT_Service simfqtService (lLogParams);
00868
00869     // DEBUG

```

```

00870     STDAIR_LOG_DEBUG ("Welcome to SimFQT display");
00871
00872     // Check whether or not a (CSV) input file should be read
00873     if (isBuiltin == true) {
00874         // Build the sample BOM tree (filled with fares) for Simfqt
00875         simfqtService.buildSampleBom();
00876     } else {
00877         // Build the BOM tree from parsing a fare file
00878         SIMFQT::FareFilePath lFareFilePath (lFareInputFilename)
;
00879         simfqtService.parseAndLoad (lFareFilePath);
00880     }
00881
00882     // DEBUG: Display the whole BOM tree
00883     const std::string& lCSVDump = simfqtService.csvDisplay();
00884     STDAIR_LOG_DEBUG (lCSVDump);
00885
00886     // DEBUG
00887     STDAIR_LOG_DEBUG ("=====");
00888     STDAIR_LOG_DEBUG ("=          Beginning of the interactive session          =");
00889     STDAIR_LOG_DEBUG ("=====");
00890
00891     // Initialise the GNU readline wrapper
00892     swift::SReadline lReader (lHistoryFilename, lHistorySize);
00893     initReadline (lReader);
00894
00895     // Now we can ask user for a line
00896     std::string lUserInput;
00897     bool EndOfInput (false);
00898     Command_T::Type_T lCommandType (Command_T::NOP);
00899
00900     while (lCommandType != Command_T::QUIT && EndOfInput == false) {
00901
00902         stdair::TravelSolutionList_T lInteractiveTravelSolutionList;
00903         stdair::TravelSolutionStruct lInteractiveTravelSolution;
00904
00905         // Update the default booking request.
00906         // If there is an input file, we want the CRS booking request (defined in
stdair).
00907         // If not, we want the default booking request.
00908         const bool isCRSBookingRequest = !isBuiltin;
00909         const stdair::BookingRequestStruct& lInteractiveBookingRequest =
simfqtService.buildBookingRequest (isCRSBookingRequest);
00910
00911         // Update the default parameters for the following interactive session.
00912         if (isBuiltin == true) {
00913             lInteractiveOrigin = "LHR";
00914             lInteractiveDestination = "SYD";
00915             lInteractiveDepartureDate = stdair::Date_T(2011,06,10);
00916             simfqtService.buildSampleTravelSolutions (lInteractiveTravelSolutionList)
;
00917
00918         } else {
00919             lInteractiveOrigin = "SIN";
00920             lInteractiveDestination = "BKK";
00921             lInteractiveDepartureDate = stdair::Date_T(2010,01,30);
00922             //
00923             const std::string lBA9_SegmentDateKey ("SQ, 970, 2010-01-30, SIN, BKK,
07:10");
00924
00925             // Add the segment date key to the travel solution.
00926             lInteractiveTravelSolution.addSegment (lBA9_SegmentDateKey);
00927
00928             // Add the travel solution to the list
00929             lInteractiveTravelSolutionList.push_back (lInteractiveTravelSolution);
00930         }
00931
00932         // Prompt.
00933         std::ostream oPromptStr;
00934         oPromptStr << "simfqt "
00935             << "> ";
00936         // The last parameter could be omitted.
00937         TokenList_T lTokenListByReadline;
00938         lUserInput = lReader.GetLine (oPromptStr.str(), lTokenListByReadline,
EndOfInput);
00939
00940         // The history could be saved to an arbitrary file at any time.
00941         lReader.SaveHistory (lHistoryBackupFilename);
00942
00943         if (EndOfInput) {
00944             std::cout << std::endl;
00945             break;
00946         }
00947
00948         // Interpret the user input.
00949         lCommandType = extractCommand (lTokenListByReadline);
00950
00951         switch (lCommandType) {
00952

```

```

00953
00954 // //////////////////////////////////// Help ////////////////////////////////////
00955 case Command_T::HELP: {
00956     // Search for information to display default parameters lists.
00957     // Get the first travel solution.
00958     stdair::TravelSolutionStruct& lTravelSolutionStruct =
00959         lInteractiveTravelSolutionList.front();
00960     // Get the segment-path of the first travel solution.
00961     const stdair::SegmentPath_T& lSegmentPath =
00962         lTravelSolutionStruct.getSegmentPath();
00963     // Get the first segment of the first travel solution.
00964     const std::string& lSegmentDateKey = lSegmentPath.front();
00965     // Get the parsed key of the first segment of the first travel solution.
00966     const stdair::ParsedKey& lParsedKey =
00967         stdair::BomKeyManager::extractKeys (lSegmentDateKey);
00968     // Get the request date time
00969     const stdair::DateTime_T& lRequestDateTime =
00970         lInteractiveBookingRequest.getRequestDateTime();
00971     const stdair::Time_T lRequestTime =
00972         lRequestDateTime.time_of_day();
00973     std::cout << std::endl;
00974     // Display help.
00975     std::cout << "Commands: " << std::endl;
00976     std::cout << " help" << "\t\t" << "Display this help" << std::endl;
00977     std::cout << " quit" << "\t\t" << "Quit the application" << std::endl;
00978     std::cout << " list" << "\t\t"
00979         << "List all the fare rule O&Ds and the corresponding date
ranges" << std::endl;
00980     std::cout << " display" << "\t"
00981         << "Display all fare rules for an O&D and a departure date. \n"
<< "\t\t"
00982         << "If no parameters specified or wrong list of parameters,
default values are used: \n" << "\t\t"
00983         << "          display " << lInteractiveOrigin << " "
00984         << lInteractiveDestination << " "
00985         << lInteractiveDepartureDate << std::endl;
00986     std::cout << " price" << "\t\t"
00987         << "Price the travel solution corresponding to a booking
request. \n" << "\t\t"
00988         << "If no parameters specified or wrong list of parameters,
default value are used: \n" << "\t\t"
00989         << "          price "
00990         << lParsedKey._airlineCode << " "
00991         << lParsedKey._flightNumber << " "
00992         << lParsedKey._departureDate << " "
00993         << lParsedKey._boardingPoint << " "
00994         << lParsedKey._offPoint << " "
00995         << lParsedKey._boardingTime << " "
00996         << lRequestDateTime.date() << " "
00997         << lRequestTime.hours() << ":" << lRequestTime.minutes() << " "
00998         << lInteractiveBookingRequest.getPOS() << " "
00999         << lInteractiveBookingRequest.getBookingChannel() << " "
01000         << lInteractiveBookingRequest.getTripType() << " "
01001         << lInteractiveBookingRequest.getStayDuration() << std::endl;
01002     std::cout << std::endl;
01003     break;
01004 }
01005
01006 // //////////////////////////////////// Quit ////////////////////////////////////
01007 case Command_T::QUIT: {
01008     break;
01009 }
01010
01011 // //////////////////////////////////// List ////////////////////////////////////
01012 case Command_T::LIST: {
01013
01014     // Get the list of all airport pairs and date ranges for which
01015     // there are fares available.
01016     const std::string& lAirportPairDateListStr =
01017         simfqtService.list ();
01018
01019     if (lAirportPairDateListStr.empty() == false) {
01020         std::cout << lAirportPairDateListStr << std::endl;
01021         STDAIR_LOG_DEBUG (lAirportPairDateListStr);
01022     } else {
01023         std::cerr << "There is no result for airport pairs and date ranges."
01024             << "Make sure your input file is not empty."
01025             << std::endl;
01026     }
01027
01028     break;
01029 }
01030
01031 // //////////////////////////////////// Display ////////////////////////////////////
01032 case Command_T::DISPLAY: {

```

```

01034
01035 // If no parameters are entered by the user, keep default ones.
01036 if (lTokenListByReadline.empty() == true) {
01037
01038     std::cout << "No parameters specified. Default paramaters '"
01039                 << lInteractiveOrigin << "-" << lInteractiveDestination
01040                 << "/" << lInteractiveDepartureDate
01041                 << "' are kept."
01042                 << std::endl;
01043
01044 } else {
01045
01046     // Find the best match corresponding to the given parameters.
01047     TokenList_T lTokenList =
01048         extractTokenListForOriDestDate (lTokenListByReadline);
01049
01050     // Parse the best match, and give default values in case the
01051     // user does not specify all the parameters or does not
01052     // specify some of them correctly.
01053     parseFlightDateKey (lTokenList, lInteractiveOrigin,
01054                        lInteractiveDestination, lInteractiveDepartureDate)
01055 ;
01056
01057 }
01058 // Check whether the selected airportpair-date is valid:
01059 // i.e. if there are corresponding fare rules.
01060 const bool isAirportPairDateValid =
01061     simfqtService.check (lInteractiveOrigin, lInteractiveDestination,
01062                        lInteractiveDepartureDate);
01063
01064 if (isAirportPairDateValid == false) {
01065     std::ostringstream oFDKStr;
01066     oFDKStr << "The airport pair/departure date: "
01067             << lInteractiveOrigin << "-" << lInteractiveDestination
01068             << "/" << lInteractiveDepartureDate
01069             << " does not correpond to any fare rule.\n"
01070             << "Make sure it exists with the 'list' command.";
01071     std::cout << oFDKStr.str() << std::endl;
01072     STDAIR_LOG_ERROR (oFDKStr.str());
01073
01074     break;
01075 }
01076
01077 // Display the list of corresponding fare rules.
01078 std::cout << "List of fare rules for "
01079             << lInteractiveOrigin << "-"
01080             << lInteractiveDestination << "/"
01081             << lInteractiveDepartureDate
01082             << std::endl;
01083
01084 const std::string& lFareRuleListStr =
01085     simfqtService.csvDisplay (lInteractiveOrigin,
01086                        lInteractiveDestination,
01087                        lInteractiveDepartureDate);
01088
01089 assert (lFareRuleListStr.empty() == false);
01090 std::cout << lFareRuleListStr << std::endl;
01091 STDAIR_LOG_DEBUG (lFareRuleListStr);
01092
01093 break;
01094 }
01095
01096 // ////////////////////////////////// Price //////////////////////////////////
01097 case Command_T::PRICE: {
01098
01099     // If no parameters are entered by the user, keep default ones.
01100     if (lTokenListByReadline.empty() == true) {
01101
01102         lInteractiveTravelSolution = lInteractiveTravelSolutionList.front();
01103
01104         std::cout << "No parameters specified. Default booking request and
default travel solution list are kept.\n"
01105                 << "Booking request: << "
01106                 << lInteractiveBookingRequest.display() << " >>"
01107                 << "\nTravel Solution: << "
01108                 << lInteractiveTravelSolution.display() << " >>"
01109                 << "\n***** \n"
01110                 << "Fare quote"
01111                 << "\n*****"
01112                 << std::endl;
01113
01114         // Try to fareQuote the sample list of travel solutions.
01115         try {
01116             simfqtService.quotePrices (lInteractiveBookingRequest,
01117                                     lInteractiveTravelSolutionList);
01118         } catch (stdair::ObjectNotFoundException& E) {

```



```

01119         std::cerr << "The given travel solution corresponding to the given
booking request can not be priced.\n"
01120         << E.what()
01121         << std::endl;
01122         break;
01123     }
01124 } else {
01125
01126     // Find the best match corresponding to the given parameters.
01127     TokenList_T lTokenList =
01128         extractTokenListForTSAndBR (lTokenListByReadline);
01129
01130     // Parse the best match, and give default values in case the
01131     // user does not specify all the parameters or does not
01132     // specify some of them correctly.
01133     stdair::BookingRequestStruct lFinalBookingRequest
01134         = parseTravelSolutionAndBookingRequestKey (lTokenList,
01135
01136 lInteractiveTravelSolutionList,
01137                                     lInteractiveBookingRequest
01138 );
01139
01140     assert (lInteractiveTravelSolutionList.size() >= 1);
01141     lInteractiveTravelSolution = lInteractiveTravelSolutionList.front();
01142
01143     // Display the booking request and the first travel solution
01144     // before pricing.
01145     std::cout << "Booking request: << "
01146     << lFinalBookingRequest.display() << " >>"
01147     << "\nTravel Solution: << "
01148     << lInteractiveTravelSolution.display() << " >>"
01149     << "\n***** \n"
01150     << "Fare quote"
01151     << "\n*****"
01152     << std::endl;
01153
01154     // Try to fareQuote the sample list of travel solutions.
01155     try {
01156         simfqtService.quotePrices (lFinalBookingRequest,
01157                                     lInteractiveTravelSolutionList);
01158     } catch (stdair::ObjectNotFoundException& E) {
01159         std::cerr << "The given travel solution corresponding to the given
booking request can not be priced.\n"
01160         << E.what()
01161         << std::endl;
01162         break;
01163     }
01164
01165     // Display the first travel solution after pricing:
01166     // one or more fare option have been added.
01167     lInteractiveTravelSolution = lInteractiveTravelSolutionList.front();
01168     std::cout << "Travel Solution: << "
01169     << lInteractiveTravelSolution.display() << " >>\n"
01170     << std::endl;
01171
01172     break;
01173 }
01174
01175 // /////////////////////////////////// Default / No value ///////////////////////////////////
01176 case Command_T::NOP: {
01177     break;
01178 }
01179 case Command_T::LAST_VALUE:
01180 default: {
01181     // DEBUG
01182     std::ostream oStr;
01183     oStr << "The '" << lUserInput << "' command is not yet understood.\n"
01184     << "Type help to have more information." << std::endl;
01185
01186     STDAIR_LOG_DEBUG (oStr.str());
01187     std::cout << oStr.str() << std::endl;
01188 }
01189 }
01190 }
01191
01192 // DEBUG
01193 STDAIR_LOG_DEBUG ("End of the session. Exiting.");
01194 std::cout << "End of the session. Exiting." << std::endl;
01195
01196 // Close the Log outputFile
01197 logOutputFile.close();
01198
01199 /*
01200 Note: as that program is not intended to be run on a server in
01201 production, it is better not to catch the exceptions. When it

```

```

01202     happens (that an exception is throwned), that way we get the
01203     call stack.
01204     */
01205
01206     return 0;
01207 }

```

23.61 test/simfqt/FQTTTestSuite.cpp File Reference

23.62 FQTTTestSuite.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 FQTTTestSuite
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/service/Logger.hpp>
00022 #include <stdair/bom/TravelSolutionStruct.hpp>
00023 #include <stdair/bom/BookingRequestStruct.hpp>
00024 // SimFQT
00025 #include <simfqt/SIMFQT_Service.hpp>
00026 #include <simfqt/config/simfqt-paths.hpp>
00027
00028 namespace boost_utf = boost::unit_test;
00029
00033 struct UnitTestConfig {
00035     UnitTestConfig() {
00036         static std::ofstream _test_log ("FQTTTestSuite_utfresults.xml");
00037         boost_utf::unit_test_log.set_stream (_test_log);
00038         boost_utf::unit_test_log.set_format (boost_utf::XML);
00039         boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00040         //boost_utf::unit_test_log.set_threshold_level
00041         (boost_utf::log_successful_tests);
00042     }
00043
00044     ~UnitTestConfig() {
00045     }
00046 };
00047
00048 // //////////////////////////////////////
00052 void testFareQuoterHelper (const unsigned short iTestFlag,
00053                          const stdair::Filename_T iFareInputFilename,
00054                          const bool isBuiltin) {
00055
00056     // Output log File
00057     std::ostringstream oStr;
00058     oStr << "FQTTTestSuite_" << iTestFlag << ".log";
00059     const stdair::Filename_T lLogFilename (oStr.str());
00060
00061     // Set the log parameters
00062     std::ofstream logOutputFile;
00063     // Open and clean the log outputfile
00064     logOutputFile.open (lLogFilename.c_str());
00065     logOutputFile.clear();
00066
00067     // Initialise the SimFQT service object
00068     const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG,
00069                                           logOutputFile);
00070
00071     // Initialise the Simfqt service object
00072     SIMFQT::SIMFQT_Service simfqtService (lLogParams);
00073
00074     // Check wether or not a (CSV) input file should be read
00075     if (isBuiltin == true) {
00076
00077         // Build the default sample BOM tree (filled with fares) for Simfqt
00078         simfqtService.buildSampleBom();
00079
00080     } else {
00081
00082         // Build the BOM tree from parsing the fare input file

```

```

00083     SIMFQT::FareFilePath lFareFilePath (iFareInputFilename)
00084 ;
00085     simfqtService.parseAndLoad (lFareFilePath);
00086 }
00087 // Build a sample list of travel solutions and a booking request.
00088 stdair::TravelSolutionList_T lTravelSolutionList;
00089 simfqtService.buildSampleTravelSolutions (lTravelSolutionList);
00090 stdair::BookingRequestStruct lBookingRequest =
00091     simfqtService.buildBookingRequest();
00092
00093 // Try to fareQuote the sample list of travel solutions
00094 simfqtService.quotePrices (lBookingRequest, lTravelSolutionList);
00095
00096 // Close the log file
00097 logOutputFile.close();
00098
00099 }
00100
00101 // ////////////////////////////////// Main: Unit Test Suite //////////////////////////////////
00102
00103 // Set the UTF configuration (re-direct the output to a specific file)
00104 BOOST_GLOBAL_FIXTURE (UnitTestFixture);
00105
00106 // Start the test suite
00107 BOOST_AUTO_TEST_SUITE (master_test_suite)
00108
00109
00112 BOOST_AUTO_TEST_CASE (simfqt_simple_pricing_test) {
00113
00114     // Input file name
00115     const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
00116         "/fare01.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 fareQuote the sample default list of travel solutions
00122     BOOST_CHECK_NO_THROW (testFareQuoterHelper (0, lFareInputFilename, isBuiltin)
00123 );
00124
00125 }
00126
00129 BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_01) {
00130
00131     // Input file name
00132     const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
00133         "/fareError01.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 fareQuote the sample default list of travel solutions
00139     BOOST_CHECK_THROW (testFareQuoterHelper (1, lFareInputFilename, isBuiltin),
00140         SIMFQT::AirportPairNotFoundException
00141 );
00142
00143 }
00144
00146 BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_02) {
00147
00148     // Input file name
00149     const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
00150         "/fareError02.csv");
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 fareQuote the sample default list of travel solutions
00156     BOOST_CHECK_THROW (testFareQuoterHelper (2, lFareInputFilename, isBuiltin),
00157         SIMFQT::PosOrChannelNotFoundException
00158 );
00159
00160 }
00161
00163 BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_03) {
00164
00165     // Input file name
00166     const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
00167         "/fareError03.csv");
00168
00169     // State whether the BOM tree should be built-in or parsed from an input file
00170     const bool isBuiltin = false;
00171
00172     // Try to fareQuote the sample default list of travel solutions
00173     BOOST_CHECK_THROW (testFareQuoterHelper (3, lFareInputFilename, isBuiltin),
00174         SIMFQT::FlightDateNotFoundExceptio
00175 );
00176
00177 }

```

```
00175
00180 BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_04) {
00181
00182     // Input file name
00183     const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
00184         "/fareError04.csv");
00185
00186     // State whether the BOM tree should be built-in or parsed from an input file
00187     const bool isBuiltin = false;
00188
00189     // Try to fareQuote the sample default list of travel solutions
00190     BOOST_CHECK_THROW (testFareQuoterHelper (4, lFareInputFilename, isBuiltin),
00191         SIMFQT::FlightTimeNotFoundException
00192     );
00193 }
00194
00195 BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_05) {
00196
00197     // Input file name
00198     const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
00199         "/fareError05.csv");
00200
00201     // State whether the BOM tree should be built-in or parsed from an input file
00202     const bool isBuiltin = false;
00203
00204     // Try to fareQuote the sample default list of travel solutions
00205     BOOST_CHECK_THROW (testFareQuoterHelper (5, lFareInputFilename, isBuiltin),
00206         SIMFQT::FeaturesNotFoundException
00207     );
00208 }
00209
00210 BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_06) {
00211
00212     // Input file name
00213     const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
00214         "/fareError06.csv");
00215
00216     // State whether the BOM tree should be built-in or parsed from an input file
00217     const bool isBuiltin = false;
00218
00219     // Try to fareQuote the sample default list of travel solutions
00220     BOOST_CHECK_THROW (testFareQuoterHelper (6, lFareInputFilename, isBuiltin),
00221         SIMFQT::AirlineNotFoundException
00222     );
00223 }
00224
00225 BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_07) {
00226
00227     // Input file name
00228     const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
00229         "/fareError07.csv");
00230
00231     // State whether the BOM tree should be built-in or parsed from an input file
00232     const bool isBuiltin = false;
00233
00234     // Try to fareQuote the sample default list of travel solutions
00235     BOOST_CHECK_THROW (testFareQuoterHelper (7, lFareInputFilename, isBuiltin),
00236         SIMFQT::FareFileParsingFailedException
00237     );
00238 }
00239
00240 BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_08) {
00241
00242     // Input file name
00243     const stdair::Filename_T lFareInputFilename (STDAIR_SAMPLE_DIR
00244         "/missingFile.csv");
00245
00246     // State whether the BOM tree should be built-in or parsed from an input file
00247     const bool isBuiltin = false;
00248
00249     // Try to fareQuote the sample default list of travel solutions
00250     BOOST_CHECK_THROW (testFareQuoterHelper (8, lFareInputFilename, isBuiltin),
00251         SIMFQT::FareInputFileNotFoundException
00252     );
00253 }
00254
00255 BOOST_AUTO_TEST_CASE (simfqt_error_pricing_test_09) {
00256
00257     // Input file name
00258     const stdair::Filename_T lEmptyInputFilename (STDAIR_SAMPLE_DIR
00259         "/ ");
00260
00261     // State whether the BOM tree should be built-in or parsed from an input file
00262     const bool isBuiltin = true;
00263
00264     // Try to fareQuote the sample default list of travel solutions
00265     BOOST_CHECK_NO_THROW (testFareQuoterHelper (9, lEmptyInputFilename, isBuiltin)
```

```
    );  
00275 }  
00276  
00277  
00278 // End the test suite  
00279 BOOST_AUTO_TEST_SUITE_END()  
00280  
00281
```

Index

- ~FacSimfqtServiceContext
 - SIMFQT::FacSimfqtServiceContext, [60](#)
- ~SIMFQT_Service
 - SIMFQT::SIMFQT_Service, [86](#)
- _bomRoot
 - SIMFQT::FareParserHelper::doEndFare, [58](#)
 - SIMFQT::FareParserHelper::FareRuleParser, [70](#)
- _fareRule
 - SIMFQT::FareParserHelper::doEndFare, [58](#)
 - SIMFQT::FareParserHelper::FareRuleParser, [70](#)
 - SIMFQT::FareParserHelper::ParserSemanticAction, [82](#)
 - SIMFQT::FareParserHelper::storeAdvancePurchase, [91](#)
 - SIMFQT::FareParserHelper::storeAirlineCode, [92](#)
 - SIMFQT::FareParserHelper::storeCabinCode, [93](#)
 - SIMFQT::FareParserHelper::storeChangeFees, [94](#)
 - SIMFQT::FareParserHelper::storeChannel, [96](#)
 - SIMFQT::FareParserHelper::storeClass, [97](#)
 - SIMFQT::FareParserHelper::storeDateRangeEnd, [98](#)
 - SIMFQT::FareParserHelper::storeDateRangeStart, [99](#)
 - SIMFQT::FareParserHelper::storeDestination, [101](#)
 - SIMFQT::FareParserHelper::storeEndRangeTime, [102](#)
 - SIMFQT::FareParserHelper::storeFare, [103](#)
 - SIMFQT::FareParserHelper::storeFareId, [104](#)
 - SIMFQT::FareParserHelper::storeMinimumStay, [106](#)
 - SIMFQT::FareParserHelper::storeNonRefundable, [107](#)
 - SIMFQT::FareParserHelper::storeOrigin, [108](#)
 - SIMFQT::FareParserHelper::storePOS, [110](#)
 - SIMFQT::FareParserHelper::storeSaturdayStay, [111](#)
 - SIMFQT::FareParserHelper::storeStartRangeTime, [112](#)
 - SIMFQT::FareParserHelper::storeTripType, [113](#)
- _itDay
 - SIMFQT::FareRuleStruct, [78](#)
- _itHours
 - SIMFQT::FareRuleStruct, [78](#)
- _itMinutes
 - SIMFQT::FareRuleStruct, [78](#)
- _itMonth
 - SIMFQT::FareRuleStruct, [78](#)
- _itSeconds
 - SIMFQT::FareRuleStruct, [78](#)
- _itYear
 - SIMFQT::FareRuleStruct, [78](#)
- addAirlineCode
 - SIMFQT::FareRuleStruct, [77](#)
- addClassCode
 - SIMFQT::FareRuleStruct, [77](#)
- advancePurchase
 - SIMFQT::FareParserHelper::FareRuleParser, [69](#)
- AirlineNotFoundException
 - SIMFQT::AirlineNotFoundException, [56](#)
- AirportPairNotFoundException
 - SIMFQT::AirportPairNotFoundException, [57](#)
- BINDIR
 - simfqt-paths.hpp, [156](#)
- buildBookingRequest
 - SIMFQT::SIMFQT_Service, [87](#)
- buildSampleBom
 - SIMFQT::SIMFQT_Service, [86](#)
- buildSampleTravelSolutions
 - SIMFQT::SIMFQT_Service, [87](#)
- cabinCode
 - SIMFQT::FareParserHelper::FareRuleParser, [69](#)
- calculateDate
 - SIMFQT::FareRuleStruct, [74](#)
- calculateTime
 - SIMFQT::FareRuleStruct, [74](#)
- changeFees
 - SIMFQT::FareParserHelper::FareRuleParser, [70](#)
- channel
 - SIMFQT::FareParserHelper::FareRuleParser, [69](#)
- check
 - SIMFQT::SIMFQT_Service, [88](#)
- clearAirlineCodeList
 - SIMFQT::FareRuleStruct, [77](#)
- clearClassCodeList
 - SIMFQT::FareRuleStruct, [77](#)
- CmdAbstract, [57](#)
- comments
 - SIMFQT::FareParserHelper::FareRuleParser, [68](#)
- create
 - SIMFQT::FacSimfqtServiceContext, [60](#)
- csvDisplay
 - SIMFQT::SIMFQT_Service, [88](#)
- DATADIR
 - simfqt-paths.hpp, [156](#)
- DATAROOTDIR
 - simfqt-paths.hpp, [156](#)
- DOCDIR
 - simfqt-paths.hpp, [156](#)
- date
 - SIMFQT::FareParserHelper::FareRuleParser, [69](#)
- dateRangeEnd
 - SIMFQT::FareParserHelper::FareRuleParser, [69](#)
- dateRangeStart
 - SIMFQT::FareParserHelper::FareRuleParser, [68](#)
- day_p
 - SIMFQT::FareParserHelper, [55](#)
- describe

- SIMFQT::FareRuleStruct, 75
- destination
 - SIMFQT::FareParserHelper::FareRuleParser, 68
- doEndFare
 - SIMFQT::FareParserHelper::doEndFare, 58
- doc/local/authors.doc, 114
- doc/local/codingrules.doc, 114
- doc/local/copyright.doc, 114
- doc/local/documentation.doc, 114
- doc/local/features.doc, 114
- doc/local/help_wanted.doc, 114
- doc/local/howto_release.doc, 114
- doc/local/index.doc, 114
- doc/local/installation.doc, 114
- doc/local/linking.doc, 114
- doc/local/test.doc, 114
- doc/local/users_guide.doc, 114
- doc/local/verification.doc, 114
- doc/tutorial/tutorial.doc, 115
- EXEC_PREFIX
 - simfqt-paths.hpp, 156
- FacServiceAbstract, 59
- FacSimfqtServiceContext
 - SIMFQT::FacSimfqtServiceContext, 60
 - SIMFQT::SIMFQT_ServiceContext, 89
- fare
 - SIMFQT::FareParserHelper::FareRuleParser, 70
- fare_id
 - SIMFQT::FareParserHelper::FareRuleParser, 68
- fare_key
 - SIMFQT::FareParserHelper::FareRuleParser, 68
- fare_rule
 - SIMFQT::FareParserHelper::FareRuleParser, 68
- fare_rule_end
 - SIMFQT::FareParserHelper::FareRuleParser, 68
- FareFileParser
 - SIMFQT::FareRuleGenerator, 65
- FareFileParsingFailedException
 - SIMFQT::FareFileParsingFailedException, 61
- FareFilePath
 - SIMFQT::FareFilePath, 61
- FareInputFileNotFoundExcepton
 - SIMFQT::FareInputFileNotFoundExcepton, 62
- FareParser
 - SIMFQT::FareRuleGenerator, 65
- FareParserHelper::doEndFare
 - SIMFQT::FareRuleGenerator, 65
- FareQuoteID_T
 - SIMFQT, 53
- FareRuleFileParser
 - SIMFQT::FareRuleFileParser, 64
- fareRuleGeneration
 - SIMFQT::FareParser, 63
- FareRuleParser
 - SIMFQT::FareParserHelper::FareRuleParser, 67
- FareRuleStruct
 - SIMFQT::FareRuleStruct, 72
- FeaturesNotFoundExcepton
 - SIMFQT::FeaturesNotFoundExcepton, 79
- FileNotFoundExcepton, 79
- FlightDateNotFoundExcepton
 - SIMFQT::FlightDateNotFoundExcepton, 80
- FlightTimeNotFoundExcepton
 - SIMFQT::FlightTimeNotFoundExcepton, 80
- generateFareRules
 - SIMFQT::FareRuleFileParser, 64
- getAdvancePurchase
 - SIMFQT::FareRuleStruct, 73
- getAirlineCode
 - SIMFQT::FareRuleStruct, 74
- getAirlineList
 - SIMFQT::FareRuleStruct, 74
- getAirlineListSize
 - SIMFQT::FareRuleStruct, 74
- getCabinCode
 - SIMFQT::FareRuleStruct, 73
- getChangeFees
 - SIMFQT::FareRuleStruct, 73
- getChannel
 - SIMFQT::FareRuleStruct, 73
- getClassCode
 - SIMFQT::FareRuleStruct, 74
- getClassCodeList
 - SIMFQT::FareRuleStruct, 74
- getClassCodeListSize
 - SIMFQT::FareRuleStruct, 74
- getDateRangeEnd
 - SIMFQT::FareRuleStruct, 73
- getDateRangeStart
 - SIMFQT::FareRuleStruct, 73
- getDestination
 - SIMFQT::FareRuleStruct, 72
- getFare
 - SIMFQT::FareRuleStruct, 74
- getFareID
 - SIMFQT::FareRuleStruct, 72
- getMinimumStay
 - SIMFQT::FareRuleStruct, 74
- getNonRefundable
 - SIMFQT::FareRuleStruct, 73
- getOrigin
 - SIMFQT::FareRuleStruct, 72
- getPOS
 - SIMFQT::FareRuleStruct, 73
- getSaturdayStay
 - SIMFQT::FareRuleStruct, 73
- getTimeRangeEnd
 - SIMFQT::FareRuleStruct, 73
- getTimeRangeStart
 - SIMFQT::FareRuleStruct, 73
- getTripType
 - SIMFQT::FareRuleStruct, 72
- grammar, 80
- HTMLDIR

- simfqt-paths.hpp, 156
- hour_p
 - SIMFQT::FareParserHelper, 55
- INCLUDEDIR
 - simfqt-paths.hpp, 156
- INFODIR
 - simfqt-paths.hpp, 156
- InputFilePath, 81
- instance
 - SIMFQT::FacSimfqtServiceContext, 60
- int1_p
 - SIMFQT::FareParserHelper, 54
- LIBDIR
 - simfqt-paths.hpp, 156
- LIBEXECDIR
 - simfqt-paths.hpp, 156
- list
 - SIMFQT::SIMFQT_Service, 88
- MANDIR
 - simfqt-paths.hpp, 156
- main
 - simfqt_parseFareRules.cpp, 117
- minimumStay
 - SIMFQT::FareParserHelper::FareRuleParser, 70
- minute_p
 - SIMFQT::FareParserHelper, 55
- month_p
 - SIMFQT::FareParserHelper, 55
- nonRefundable
 - SIMFQT::FareParserHelper::FareRuleParser, 70
- ObjectNotFoundException, 81
- operator<<
 - simfqt_parseFareRules.cpp, 117
- operator()
 - SIMFQT::FareParserHelper::doEndFare, 58
 - SIMFQT::FareParserHelper::storeAdvancePurchase, 90
 - SIMFQT::FareParserHelper::storeAirlineCode, 92
 - SIMFQT::FareParserHelper::storeCabinCode, 93
 - SIMFQT::FareParserHelper::storeChangeFees, 94
 - SIMFQT::FareParserHelper::storeChannel, 95
 - SIMFQT::FareParserHelper::storeClass, 97
 - SIMFQT::FareParserHelper::storeDateRangeEnd, 98
 - SIMFQT::FareParserHelper::storeDateRangeStart, 99
 - SIMFQT::FareParserHelper::storeDestination, 100
 - SIMFQT::FareParserHelper::storeEndRangeTime, 102
 - SIMFQT::FareParserHelper::storeFare, 103
 - SIMFQT::FareParserHelper::storeFareId, 104
 - SIMFQT::FareParserHelper::storeMinimumStay, 105
 - SIMFQT::FareParserHelper::storeNonRefundable, 107
- SIMFQT::FareParserHelper::storeOrigin, 108
- SIMFQT::FareParserHelper::storePOS, 109
- SIMFQT::FareParserHelper::storeSaturdayStay, 111
- SIMFQT::FareParserHelper::storeStartRangeTime, 112
- SIMFQT::FareParserHelper::storeTripType, 113
- origin
 - SIMFQT::FareParserHelper::FareRuleParser, 68
- PACKAGE
 - simfqt-paths.hpp, 155
- PACKAGE_NAME
 - simfqt-paths.hpp, 155
- PACKAGE_VERSION
 - simfqt-paths.hpp, 155
- PDFDIR
 - simfqt-paths.hpp, 156
- PREFIXDIR
 - simfqt-paths.hpp, 156
- parseAndLoad
 - SIMFQT::SIMFQT_Service, 86
- ParserSemanticAction
 - SIMFQT::FareParserHelper::ParserSemanticAction, 82
- ParsingFileFailedException, 83
- point_of_sale
 - SIMFQT::FareParserHelper::FareRuleParser, 69
- PosOrChannelNotFoundException
 - SIMFQT::PosOrChannelNotFoundException, 84
- quotePrices
 - SIMFQT::SIMFQT_Service, 87
- readConfiguration
 - simfqt_parseFareRules.cpp, 117
- RootException, 84
- SBINDIR
 - simfqt-paths.hpp, 156
- SIMFQT, 53
 - FareQuoteID_T, 53
 - SIMFQT_ServicePtr_T, 53
- SIMFQT::AirlineNotFoundException, 56
 - AirlineNotFoundException, 56
- SIMFQT::AirportPairNotFoundExceptio, 56
 - AirportPairNotFoundExceptio, 57
- SIMFQT::FacSimfqtServiceContext, 59
 - ~FacSimfqtServiceContext, 60
 - create, 60
 - FacSimfqtServiceContext, 60
 - instance, 60
- SIMFQT::FareFileParsingFailedException, 60
 - FareFileParsingFailedException, 61
- SIMFQT::FareFilePath, 61
 - FareFilePath, 61
- SIMFQT::FareInputFileNotFoundExceptio, 62
 - FareInputFileNotFoundExceptio, 62
- SIMFQT::FareParser, 62

- fareRuleGeneration, 63
- SIMFQT::FareParserHelper, 54
 - day_p, 55
 - hour_p, 55
 - int1_p, 54
 - minute_p, 55
 - month_p, 55
 - second_p, 55
 - uint1_4_p, 55
 - uint2_p, 54
 - uint4_p, 55
 - year_p, 55
- SIMFQT::FareParserHelper::FareRuleParser
 - _bomRoot, 70
 - _fareRule, 70
 - advancePurchase, 69
 - cabinCode, 69
 - changeFees, 70
 - channel, 69
 - comments, 68
 - date, 69
 - dateRangeEnd, 69
 - dateRangeStart, 68
 - destination, 68
 - fare, 70
 - fare_id, 68
 - fare_key, 68
 - fare_rule, 68
 - fare_rule_end, 68
 - FareRuleParser, 67
 - minimumStay, 70
 - nonRefundable, 70
 - origin, 68
 - point_of_sale, 69
 - saturdayStay, 70
 - segment, 70
 - start, 68
 - time, 69
 - timeRangeEnd, 69
 - timeRangeStart, 69
 - tripType, 68
- SIMFQT::FareParserHelper::FareRuleParser< Iterator>, 65
- SIMFQT::FareParserHelper::ParserSemanticAction, 81
 - _fareRule, 82
 - ParserSemanticAction, 82
- SIMFQT::FareParserHelper::doEndFare, 57
 - _bomRoot, 58
 - _fareRule, 58
 - doEndFare, 58
 - operator(), 58
- SIMFQT::FareParserHelper::storeAdvancePurchase, 90
 - _fareRule, 91
 - operator(), 90
 - storeAdvancePurchase, 90
- SIMFQT::FareParserHelper::storeAirlineCode, 91
 - _fareRule, 92
 - operator(), 92
 - storeAirlineCode, 91
- SIMFQT::FareParserHelper::storeCabinCode, 92
 - _fareRule, 93
 - operator(), 93
 - storeCabinCode, 93
- SIMFQT::FareParserHelper::storeChangeFees, 93
 - _fareRule, 94
 - operator(), 94
 - storeChangeFees, 94
- SIMFQT::FareParserHelper::storeChannel, 95
 - _fareRule, 96
 - operator(), 95
 - storeChannel, 95
- SIMFQT::FareParserHelper::storeClass, 96
 - _fareRule, 97
 - operator(), 97
 - storeClass, 96
- SIMFQT::FareParserHelper::storeDateRangeEnd, 97
 - _fareRule, 98
 - operator(), 98
 - storeDateRangeEnd, 98
- SIMFQT::FareParserHelper::storeDateRangeStart, 98
 - _fareRule, 99
 - operator(), 99
 - storeDateRangeStart, 99
- SIMFQT::FareParserHelper::storeDestination, 100
 - _fareRule, 101
 - operator(), 100
 - storeDestination, 100
- SIMFQT::FareParserHelper::storeEndRangeTime, 101
 - _fareRule, 102
 - operator(), 102
 - storeEndRangeTime, 102
- SIMFQT::FareParserHelper::storeFare, 102
 - _fareRule, 103
 - operator(), 103
 - storeFare, 103
- SIMFQT::FareParserHelper::storeFareId, 104
 - _fareRule, 104
 - operator(), 104
 - storeFareId, 104
- SIMFQT::FareParserHelper::storeMinimumStay, 105
 - _fareRule, 106
 - operator(), 105
 - storeMinimumStay, 105
- SIMFQT::FareParserHelper::storeNonRefundable, 106
 - _fareRule, 107
 - operator(), 107
 - storeNonRefundable, 107
- SIMFQT::FareParserHelper::storeOrigin, 107
 - _fareRule, 108
 - operator(), 108
 - storeOrigin, 108
- SIMFQT::FareParserHelper::storePOS, 109
 - _fareRule, 110
 - operator(), 109
 - storePOS, 109
- SIMFQT::FareParserHelper::storeSaturdayStay, 110

[_fareRule, 111](#)
[operator\(\), 111](#)
[storeSaturdayStay, 110](#)
SIMFQT::FareParserHelper::storeStartRangeTime, [111](#)
[_fareRule, 112](#)
[operator\(\), 112](#)
[storeStartRangeTime, 112](#)
SIMFQT::FareParserHelper::storeTripType, [112](#)
[_fareRule, 113](#)
[operator\(\), 113](#)
[storeTripType, 113](#)
SIMFQT::FareQuoter, [63](#)
[SIMFQT_Service, 63](#)
SIMFQT::FareRuleFileParser, [63](#)
[FareRuleFileParser, 64](#)
[generateFareRules, 64](#)
SIMFQT::FareRuleGenerator, [64](#)
[FareFileParser, 65](#)
[FareParser, 65](#)
[FareParserHelper::doEndFare, 65](#)
SIMFQT::FareRuleStruct, [71](#)
[_itDay, 78](#)
[_itHours, 78](#)
[_itMinutes, 78](#)
[_itMonth, 78](#)
[_itSeconds, 78](#)
[_itYear, 78](#)
[addAirlineCode, 77](#)
[addClassCode, 77](#)
[calculateDate, 74](#)
[calculateTime, 74](#)
[clearAirlineCodeList, 77](#)
[clearClassCodeList, 77](#)
[describe, 75](#)
[FareRuleStruct, 72](#)
[getAdvancePurchase, 73](#)
[getAirlineCode, 74](#)
[getAirlineList, 74](#)
[getAirlineListSize, 74](#)
[getCabinCode, 73](#)
[getChangeFees, 73](#)
[getChannel, 73](#)
[getClassCode, 74](#)
[getClassCodeList, 74](#)
[getClassCodeListSize, 74](#)
[getDateRangeEnd, 73](#)
[getDateRangeStart, 73](#)
[getDestination, 72](#)
[getFare, 74](#)
[getFareID, 72](#)
[getMinimumStay, 74](#)
[getNonRefundable, 73](#)
[getOrigin, 72](#)
[getPOS, 73](#)
[getSaturdayStay, 73](#)
[getTimeRangeEnd, 73](#)
[getTimeRangeStart, 73](#)
[getTripType, 72](#)
[setAdvancePurchase, 76](#)
[setAirlineCode, 77](#)
[setCabinCode, 76](#)
[setChangeFees, 76](#)
[setChannel, 76](#)
[setClassCode, 77](#)
[setDateRangeEnd, 75](#)
[setDateRangeStart, 75](#)
[setDestination, 75](#)
[setFare, 77](#)
[setFareID, 75](#)
[setMinimumStay, 77](#)
[setNonRefundable, 76](#)
[setOrigin, 75](#)
[setPOS, 76](#)
[setSaturdayStay, 76](#)
[setTimeRangeEnd, 76](#)
[setTimeRangeStart, 75](#)
[setTripType, 75](#)
SIMFQT::FeaturesNotFoundException, [78](#)
[FeaturesNotFoundException, 79](#)
SIMFQT::FlightDateNotFoundException, [79](#)
[FlightDateNotFoundException, 80](#)
SIMFQT::FlightTimeNotFoundException, [80](#)
[FlightTimeNotFoundException, 80](#)
SIMFQT::PosOrChannelNotFoundException, [83](#)
[PosOrChannelNotFoundException, 84](#)
SIMFQT::QuotingException, [84](#)
SIMFQT::SIMFQT_Service, [85](#)
[buildBookingRequest, 87](#)
[buildSampleBom, 86](#)
[buildSampleTravelSolutions, 87](#)
[check, 88](#)
[csvDisplay, 88](#)
[list, 88](#)
[parseAndLoad, 86](#)
[quotePrices, 87](#)
SIMFQT::SIMFQT_ServiceContext, [89](#)
[FacSimfqtServiceContext, 89](#)
SIMFQT_Service
[SIMFQT::FareQuoter, 63](#)
[SIMFQT::SIMFQT_Service, 85, 86](#)
[SIMFQT::SIMFQT_ServiceContext, 89](#)
SIMFQT_ServicePtr_T
[SIMFQT, 53](#)
STDAIR_SAMPLE_DIR
[simfqt-paths.hpp, 157](#)
SYSCONFDIR
[simfqt-paths.hpp, 156](#)
saturdayStay
[SIMFQT::FareParserHelper::FareRuleParser, 70](#)
second_p
[SIMFQT::FareParserHelper, 55](#)
segment
[SIMFQT::FareParserHelper::FareRuleParser, 70](#)
ServiceAbstract, [85](#)
setAdvancePurchase
[SIMFQT::FareRuleStruct, 76](#)

- setAirlineCode
 - SIMFQT::FareRuleStruct, 77
- setCabinCode
 - SIMFQT::FareRuleStruct, 76
- setChangeFees
 - SIMFQT::FareRuleStruct, 76
- setChannel
 - SIMFQT::FareRuleStruct, 76
- setClassCode
 - SIMFQT::FareRuleStruct, 77
- setDateRangeEnd
 - SIMFQT::FareRuleStruct, 75
- setDateRangeStart
 - SIMFQT::FareRuleStruct, 75
- setDestination
 - SIMFQT::FareRuleStruct, 75
- setFare
 - SIMFQT::FareRuleStruct, 77
- setFareID
 - SIMFQT::FareRuleStruct, 75
- setMinimumStay
 - SIMFQT::FareRuleStruct, 77
- setNonRefundable
 - SIMFQT::FareRuleStruct, 76
- setOrigin
 - SIMFQT::FareRuleStruct, 75
- setPOS
 - SIMFQT::FareRuleStruct, 76
- setSaturdayStay
 - SIMFQT::FareRuleStruct, 76
- setTimeRangeEnd
 - SIMFQT::FareRuleStruct, 76
- setTimeRangeStart
 - SIMFQT::FareRuleStruct, 75
- setTripType
 - SIMFQT::FareRuleStruct, 75
- simfqt-paths.hpp
 - BINDIR, 156
 - DATADIR, 156
 - DATAROOTDIR, 156
 - DOCDIR, 156
 - EXEC_PREFIX, 156
 - HTMLDIR, 156
 - INCLUDEDIR, 156
 - INFODIR, 156
 - LIBDIR, 156
 - LIBEXECDIR, 156
 - MANDIR, 156
 - PACKAGE, 155
 - PACKAGE_NAME, 155
 - PACKAGE_VERSION, 155
 - PDFDIR, 156
 - PREFIXDIR, 156
 - SBINDIR, 156
 - STDAIR_SAMPLE_DIR, 157
 - SYSCONFDIR, 156
- simfqt/SIMFQT_Service.hpp, 166, 167
- simfqt/SIMFQT_Types.hpp, 168, 169
- simfqt/basic/BasConst.cpp, 115
- simfqt/basic/BasConst_General.hpp, 115
- simfqt/basic/BasConst_SIMFQT_Service.hpp, 115, 116
- simfqt/batches/simfqt_parseFareRules.cpp, 116, 118
- simfqt/bom/FareRuleStruct.cpp, 120, 121
- simfqt/bom/FareRuleStruct.hpp, 122
- simfqt/command/FareParser.cpp, 126
- simfqt/command/FareParser.hpp, 127
- simfqt/command/FareParserHelper.cpp, 127, 128
- simfqt/command/FareParserHelper.hpp, 137, 138
- simfqt/command/FareQuoter.cpp, 140, 141
- simfqt/command/FareQuoter.hpp, 149
- simfqt/command/FareRuleGenerator.cpp, 150, 151
- simfqt/command/FareRuleGenerator.hpp, 154
- simfqt/config/simfqt-paths.hpp, 155, 157
- simfqt/factory/FacSimfqtServiceContext.cpp, 157
- simfqt/factory/FacSimfqtServiceContext.hpp, 158
- simfqt/service/SIMFQT_Service.cpp, 159
- simfqt/service/SIMFQT_ServiceContext.cpp, 164
- simfqt/service/SIMFQT_ServiceContext.hpp, 165
- simfqt/ui/cmdline/simfqt.cpp, 170
- simfqt_parseFareRules.cpp
 - main, 117
 - operator<<, 117
 - readConfiguration, 117
 - WordList_T, 117
- start
 - SIMFQT::FareParserHelper::FareRuleParser, 68
- stdair, 55
- storeAdvancePurchase
 - SIMFQT::FareParserHelper::storeAdvancePurchase, 90
- storeAirlineCode
 - SIMFQT::FareParserHelper::storeAirlineCode, 91
- storeCabinCode
 - SIMFQT::FareParserHelper::storeCabinCode, 93
- storeChangeFees
 - SIMFQT::FareParserHelper::storeChangeFees, 94
- storeChannel
 - SIMFQT::FareParserHelper::storeChannel, 95
- storeClass
 - SIMFQT::FareParserHelper::storeClass, 96
- storeDateRangeEnd
 - SIMFQT::FareParserHelper::storeDateRangeEnd, 98
- storeDateRangeStart
 - SIMFQT::FareParserHelper::storeDateRangeStart, 99
- storeDestination
 - SIMFQT::FareParserHelper::storeDestination, 100
- storeEndRangeTime
 - SIMFQT::FareParserHelper::storeEndRangeTime, 102
- storeFare
 - SIMFQT::FareParserHelper::storeFare, 103
- storeFareId
 - SIMFQT::FareParserHelper::storeFareId, 104
- storeMinimumStay

- SIMFQT::FareParserHelper::storeMinimumStay,
[105](#)
- storeNonRefundable
 - SIMFQT::FareParserHelper::storeNonRefundable,
[107](#)
- storeOrigin
 - SIMFQT::FareParserHelper::storeOrigin, [108](#)
- storePOS
 - SIMFQT::FareParserHelper::storePOS, [109](#)
- storeSaturdayStay
 - SIMFQT::FareParserHelper::storeSaturdayStay,
[110](#)
- storeStartRangeTime
 - SIMFQT::FareParserHelper::storeStartRange-
Time, [112](#)
- storeTripType
 - SIMFQT::FareParserHelper::storeTripType, [113](#)
- StructAbstract, [114](#)
- test/simfqt/FQTTTestSuite.cpp, [184](#)
- time
 - SIMFQT::FareParserHelper::FareRuleParser, [69](#)
- timeRangeEnd
 - SIMFQT::FareParserHelper::FareRuleParser, [69](#)
- timeRangeStart
 - SIMFQT::FareParserHelper::FareRuleParser, [69](#)
- tripType
 - SIMFQT::FareParserHelper::FareRuleParser, [68](#)
- uint1_4_p
 - SIMFQT::FareParserHelper, [55](#)
- uint2_p
 - SIMFQT::FareParserHelper, [54](#)
- uint4_p
 - SIMFQT::FareParserHelper, [55](#)
- WordList_T
 - simfqt_parseFareRules.cpp, [117](#)
- year_p
 - SIMFQT::FareParserHelper, [55](#)