Horus
IMSETY
Detailed Design Document
Version 0.1 21st June 2007

Project team:
Jeroen Keiren 0569081
Frank Koenders 0575629
Thijs Nugteren 0574426
Joeri de Ruiter 0578312
Stijn Stiefelhagen 0579816
Carst Tankink 0569954
Pim Vullers 0575766
Freek van Walderveen 0566348

Project manager:
Egbert Teeselink

Senior management: L. Somers TU/e (HG 7.83)
M. v.d. Brand TU/e (HG 7.44)

Adviser: R.J. Bril TU/e (HG 5.09)

Customer: E. v. Breukelen ISIS

Computer Science, Eindhoven University of Technology, Eindhoven
Abstract

This document contains the Detailed Design for IMSETY. The document complies with the Detailed Design Document (DDD) from the Software Engineering Standard as defined by the European Space Agency. It defines several conventions and standards concerning documents and code. Furthermore it describes all components in the system, the build procedure and traces the software requirements from the SRD [5] to the components in the DDD.
Contents

1 Introduction ............................................. 1
  1.1 Purpose ............................................ 1
  1.2 Scope ............................................. 1
  1.3 List of definitions .................................. 1
  1.4 List of references .................................. 1
  1.5 Overview ......................................... 2

2 Standards and conventions ......................... 3
  2.1 Design standards ................................... 3
  2.2 Documentation standards ........................... 3
  2.3 Naming conventions ................................ 3
  2.4 Coding standards ................................. 3
  2.5 Software development tools ......................... 3

3 Component descriptions ............................. 4
  3.1 Client ............................................ 4
  3.2 Server ........................................... 107

4 Build procedure ....................................... 421
  4.1 Dependencies ...................................... 421
  4.2 Server ........................................... 423
  4.3 Client ........................................... 424

A Source code listings .................................. 425

B Requirements traceability matrix ................. 426
## Document status sheet

<table>
<thead>
<tr>
<th>Document title</th>
<th>Detailed Design Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document identifier</td>
<td>IMSETY/doc/DDD</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Jeroen Keiren, Frank Koenders, Thijs Nugteren, Joeri de Ruiter, Stijn Stiefelhagen, Carst Tankink, Pim Vullers, Freek van Walderveen</td>
</tr>
<tr>
<td>Version</td>
<td>0.1</td>
</tr>
<tr>
<td>Document status</td>
<td>Internally approved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author(s)</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1 (Revision 2000)</td>
<td>21-06-2007</td>
<td>Jeroen Keiren, Frank Koenders, Thijs Nugteren, Joeri de Ruiter, Stijn Stiefelhagen, Carst Tankink, Pim Vullers, Freek van Walderveen</td>
<td>Documentation concerning IMSETY version 0.2.1</td>
</tr>
</tbody>
</table>
### Document change report

<table>
<thead>
<tr>
<th><strong>Document title</strong></th>
<th>Detailed Design Document</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Document identifier</strong></td>
<td>IMSETY/doc/DDD</td>
</tr>
<tr>
<td><strong>Date of changes</strong></td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Section number</strong></th>
<th><strong>Reason for change</strong></th>
</tr>
</thead>
</table>
Chapter 1

Introduction

1.1 Purpose

The DDD describes each component in the system and shows the connection between these components. It is meant for developers who wish to get an understanding of the system.

1.2 Scope

IMSETY will provide scientists and observers with an easy to use program interface to conduct experiments on space based samples and possible reference samples on Earth.

1.3 List of definitions

This section contains the definitions of all used terms, acronyms and abbreviations in this document.

ADD Architectural Design Document
DBMS Database Management System
DDD Detailed Design Document
ESA European Space Agency
GENSO Global Educational Network for Satellite Operations
MCC Mission Control Client, see the URD [6] for more information.
MCCC Mission Control Client Controller
MCS Mission Control Software, see the URD [6] for more information.
MCSC Mission Control Software Connectivity
SQAP Software Quality Assurance Plan
SRD Software Requirements Document

1.4 List of references

1.5 Overview

The remainder of this document is divided into the following chapters, as dictated by the ESA Software Engineering Standards [1]: Standards and conventions (chapter 2) describes the standards and conventions used in the IMSETY project. After this all components in the system will be described in Component description (chapter 3). Build procedure (chapter 4) will explain how to build the entire system from the source files. In the source code listings (Appendix A) a listing of source code files will be shown per component for both server and client. And finally the Requirements traceability matrix (Appendix B) will link each software requirement of the SRD to the components in the DDD.
Chapter 2

Standards and conventions

2.1 Design standards

In producing the IMSETY system we have used the Object Oriented design method in order to make a natural decomposition of the system into components.

2.2 Documentation standards

For the automatic generation of the code documentation we have used Doxygen. Doxygen can be found at http://www.doxygen.org.

2.3 Naming conventions

The file names of the source code generally reflect their function within the system.

2.4 Coding standards

The coding standard are used as described in the SQAP [4].

2.5 Software development tools

For the server we have used SCons as build environment. For the client it is more convenient to use Make, because the Makefile is automatically generated by Qt. Qt designer was used in building the GUI for the client.
Chapter 3

Component descriptions

3.1 Client

3.1.1 Future work

Administration

In order to implement an administration front end in the client, one should implement the desired functionality in the server. This includes implementing the stubs still left in the controller component, and adding XML-RPC functions to the client connectivity component.

To use the functionality provided by the server, the following needs to be added to the client:

User interface Several views are available for administration. These need to be extended with the fields in the database that can actually be edited by an administrator. These fields are the arguments of the function call in the server’s controller.

Connectivity - Interface connection In order to connect the implemented connectivity to the user interface, we advise to use a model-view-controller architecture, as implemented by Qt. An example of this use can be seen in our implementation of the experiment view and the corresponding functionality in the server connectivity.

A part of the administration which can be done by scientists, is the observer management. This functionality is different from the standard administration, and therefore we describe it separately here.

In order to implement observer management, getAccounts and grantRights functions need to be added to the controller in the server. These functions can be used by administrators and scientist to grant other accounts observation rights to experiments. These functions also need to be added to the client connectivity.

Now, to manage observers, the manage observers dialog in the client user interface needs to be coupled, through the server connectivity, to the functions in the server. It should present the user with a list of accounts, retrieved via the getAccounts function. After the observers have been added, they can observe the experiment to which they have been granted the rights.
CHAPTER 3. COMPONENT DESCRIPTIONS

Scheduling

In the scheduling dialog, the main discrepancy is a lacking visualization of the timeline, which is hinted at by the stub in the scheduling window. The functionality we had in mind with this timeline was the following:

1. The list of opportunities list box is sorted according to how much of the experiment is covered by the opportunities.
2. The user selects the opportunities during which she wants to observe the experiment.
3. These opportunities are visualized to scale, on the timeline to the right of the list box. The current timeline is a stub, and should be replaced by a QWidget descendant which needs to be implemented.
4. Next to the opportunities, the experiment is also visualized, to scale.
5. The user can drag the experiment so it is in line with the selected opportunities. In the mean time, the time edit field “Start time” is updated accordingly.
6. The user can tweak the start time using the time edit.
7. When she is satisfied, the user clicks the “Schedule” button.

The rightmost list box, under “Experiments scheduled” can be used to display detailed information about a selected opportunity. This opportunity can either be clicked in the timeline visualization or in the opportunities list box. This information can, among others, comprise other experiments scheduled at this opportunity, including the names of the scientists which scheduled those experiments.

Intervention

To implement the intervention, the sendCommand function in the controller needs to be implemented in the controller and added to the client connectivity.

To implement the intervention functionality in the client, one can display the list of commands of the experiment in the sendCommandsDockWidget of the “Observation and intervention” dialog. The “Send” button can then be linked to a function call in the server.

Observation data manipulation

The a posteriori manipulation of observation data contains the following functions, shown in the context menu of thumbnails in the observation data dialog. These functions are:

Mark interesting This function marks an observation as interesting. This is a local marking, which allows a scientist to later on show only interesting observations and quickly download high resolution data for all interesting observations.

Mark for deletion This function requests the server to upload a command to the satellite that deletes an observation and all associated files on the satellite. It might be desirable to show a warning to the user before uploading the command, since it is a destructive command.

Request hi-res This function requests the server to download a high resolution version of an observation from the satellite. This function should only be enabled when a high resolution image is actually available on the satellite, and it should be made clear to the user that this might take some time.
Show hi-res This function opens an “Observation” dialog with the high resolution version of the observation. This function should only be enabled when a high resolution version of the data is available on the server.

For the thumbnail overview, it might be desirable to implement functionality which streams the thumbnails one by one to the client and displays them as they come in. At the moment, all thumbnails are requested at once, leading to a large delay. Also, more information per thumbnail would need to be sent to make it useful.

Furthermore, currently one can only filter on the experiment to which observations are associated. We think implementing more filters could be useful.

Plug-in system

The plug-in system still lacks a detailed design. The general idea modelled in the ADD [3] can be followed. This means that when data is retrieved from the server, it is first fetched to a processor batch, which is filled with plug-ins conforming to an interface, which is also shown in the ADD.

The plug-in framework itself can be implemented using the Qt plug-in framework [2].

3.1.2 IMSETY client Directories

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

client .................................................. 9

3.1.3 IMSETY client Class Hierarchy

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

Experiment ........................................... 21
Opportunity ......................................... 52
PossibleCommand .................................. 53
QDialog ............................................. 56
CommandsEditDialog ............................... 15
CommunicationLogsDialog ....................... 16
CreateExperimentDialog ......................... 17
LoginDialog ....................................... 29
ManageObservers ................................ 38
ManagePayloadsDialog ......................... 40
ManageSatellitesDialog ......................... 41
ManageUsersDialog ............................... 42
ObservationDataDialog ......................... 45
Preferences ...................................... 54
SelectObserveDialog ............................. 60
QMainWindow ...................................... 56
ExperimentDialog ................................ 21
MainWindow ....................................... 33
Observe ........................................... 50
QObject ........................................... 57
MultiHttp .......................................... 43
ServerConnectivity .............................. 61
CHAPTER 3. COMPONENT DESCRIPTIONS

QStandardItemModel ........................................... 59
UserModel ..................................................... 77
QWidget .......................................................... 59
Commands ......................................................... 10
ObservationView ............................................... 47
UserSelectionWidget .......................................... 77
Thumbnail ....................................................... 75
TimedCommand .................................................. 76

3.1.4 IMSETY client Class List

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

Commands (A command combined with a time) .................. 10
CommandsEditDialog (The dialog in which the commands can be edited) ............ 15
CommunicationLogsDialog (The part of the user interface in which communication logs can be viewed) ................... 16
CreateExperimentDialog (The part of the user interface which handles the communication logs) .................. 17
Experiment ..................................................... 21
ExperimentDialog ............................................... 21
LoginDialog (Needed for the log in animation) ...................... 29
MainWindow (Needed for the model view controller) ................. 33
ManageObservers .............................................. 38
ManagePayloadsDialog .......................................... 40
ManageSatellitesDialog .......................................... 41
ManageUsersDialog ............................................. 42
MultiHttp ....................................................... 43
ObservationDataDialog ......................................... 45
ObservationView .................................................. 47
Observe (Provides functionality for observing an experiment to the client) ........... 50
Opportunity ..................................................... 52
PossibleCommand .............................................. 53
Preferences ..................................................... 54
QDialog ........................................................ 56
QMainWindow .................................................... 56
QObject .......................................................... 57
QStandardItemModel ............................................. 59
QWidget .......................................................... 59
SelectObserveDialog ............................................. 60
ServerConnectivity ............................................. 60
Thumbnail ....................................................... 75
TimedCommand .................................................. 76
UserModel ........................................................ 77
UserSelectionWidget ............................................. 77

3.1.5 IMSETY client File List

Here is a list of all files with brief descriptions:

client/commands.cpp ............................................. 79
client/commands.h .............................................. 80
3.1.6 IMSETY client Related Pages

Here is a list of all related documentation pages:

Todo List ............................................. 105
CHAPTER 3. COMPONENT DESCRIPTIONS

3.1.7 client/ Directory Reference

Files

- file `commands.cpp`
- file `commands.h`
- file `commandseditdialog.cpp`
- file `commandseditdialog.h`
- file `communicationlogsdialog.cpp`
- file `communicationlogsdialog.h`
- file `conntest.cpp`
- file `createexperimentdialog.cpp`
- file `createexperimentdialog.h`
- file `experimentdialog.cpp`
- file `experimentdialog.h`
- file `logindialog.cpp`
- file `logindialog.h`
- file `main.cpp`
- file `mainwindow.cpp`
- file `mainwindow.h`
- file `manageobservers.cpp`
- file `manageobservers.h`
- file `managepayloadsdialog.cpp`
- file `managepayloadsdialog.h`
- file `managesatellitesdialog.cpp`
- file `managesatellitesdialog.h`
- file `manageusersdialog.cpp`
- file `manageusersdialog.h`
- file `multihttp.cpp`
- file `multihttp.h`
- file `objects.h`
- file `observationdatadialog.cpp`
- file `observationdatadialog.h`
- file `observationview.cpp`
- file `observationview.h`
- file `observe.cpp`
- file `observe.h`
- file `preferences.cpp`
- file `preferences.h`
- file `selectobservedialog.cpp`
- file `selectobservedialog.h`
- file `serverconnectivity.cpp`
- file `serverconnectivity.h`
- file `usermodel.h`
- file `userselectionwidget.cpp`
- file `userselectionwidget.h`
- file `version.h`
### 3.1.8 Commands Class Reference

A command combined with a time.

```cpp
#include <commands.h>
```

Inherits `QWidget`.

Inheritance diagram for `Commands`:

![Inheritance diagram](image)

**Signals**

- void `selectionChanged (bool hasSelection)`
  
  Signal used to indicate if the selection has changed.

**Public Member Functions**

- void `addCommand (const PossibleCommand &cmd)`
  
  Adds a command to the model.

- void `clearSelection ()`
  
  Clears the selection from the model.

- `Commands (QWidget *parent=0)`
  
  Constructor of a `Commands` (p. 10) object.

- bool `getCommand (TimedCommand *cmd)`
  
  Gets a certain command.

- `QString getCommandName (int commandId)`
  
  Get the name given a commandID.

- void `selectCommand (const TimedCommand &cmd)`
Selects a command from the model.

- \∼ Commands ()
  Destructor of Commands (p. 10) object.

Private Types

- enum { ParamDefaultRole = Qt::UserRole + 1, ParamMinRole, ParamMaxRole, IdRole }
  Enumerator of different roles.

Private Slots

- void commandList\_currentChanged ()
  Slot used to update the parameter list.

Private Member Functions

- void addParameter (const QString &parameter, const QVariant &defaultValue, const QVariant &min, const QVariant &max)
  Adds a parameter to the parameter list belonging to a command.

- void setupCommands ()
  Set up the command model.

- void setupParameterHeader ()
  Set the header labels of the parameter list.

- void setupParameters ()
  Set up the parameter model.

Private Attributes

- QStandardItemModel * commandModel
- QStandardItemModel * parameterModel
  Needed for the model view controller.

- Ui::Commands \texttt{ui}

Member Enumeration Documentation

anonymous enum [private]

Enumerator:

- ParamDefaultRole
**Chapter 3. Component Descriptions**

*ParamMinRole*

*ParamMaxRole*

*IdRole*

**Constructor & Destructor Documentation**

**Commands** (QWidget ∗ *parent* = 0)

**Parameters:**

*parent* A QWidget (p. 59) representing the parent window.

**Precondition:**

true

**Postcondition:**

Commands (p. 10) and parameters are set up.

**~Commands** ()

**Member Function Documentation**

void addCommand (const PossibleCommand & *cmd*)

**Parameters:**

*cmd* Representing a possible command.

**Precondition:**

true

**Postcondition:**

The command is added to the model.

void addParameter (const QString & *parameter*, const QVariant & *defaultValue*, const QVariant & *min*, const QVariant & *max*) [private]

**Parameters:**

*parameter* The name of the parameter.

*defaultValue* The default value of a parameter.

*min* The minimal value of the parameter.

*max* The maximal value of the parameter.

**Precondition:**

true

**Postcondition:**

The parameter is added to the parameterlist belonging to a command.

---

12 DDD 0.1
void clearSelection ()

Precondition:
  true

Postcondition:
  The selection in the model is cleared.

void commandList_currentChanged () [private, slot]

Precondition:
  true

Postcondition:
  The parameter list is updated.

bool getCommand (TimedCommand * cmd)

Parameters:
  cmd A TimedCommand (p. 76) representing a command in the experiment.

Precondition:
  true

Returns:
  Whether the command cmd is selected or not.

QString getCommandName (int commandId)

Parameters:
  commandId Identifying a command.

Precondition:
  commandId > -1

Returns:
  The name of the command.
void selectCommand (const TimedCommand & cmd)

Parameters:
  
  cmd Representing a TimedCommand (p. 76).

Precondition:
  
  true

Postcondition:
  The command is selected in the model.

void selectionChanged (bool hasSelection) [signal]

void setupCommands () [private]

Precondition:
  
  true

Postcondition:
  The command model is set up.

Todo
  Sort commands

void setupParameterHeader () [private]

Precondition:
  
  true

Postcondition:
  The headers of the parameterlist are set.

void setupParameters () [private]

Precondition:
  
  true

Postcondition:
  The parameter model is set up.

Member Data Documentation

QStandardItemModel * commandModel [private]
CHAPTER 3. COMPONENT DESCRIPTIONS

QStandardItemModel* parameterModel  [private]

Ui::Commands ui  [private]
Used for the user interface.
The documentation for this class was generated from the following files:

- client/commands.h
- client/commands.cpp

3.1.9 CommandsEditDialog Class Reference

The dialog in which the commands can be edited.

#include <commandseditdialog.h>

Inherits QDialog.

Inheritance diagram for CommandsEditDialog:

Public Member Functions

- CommandsEditDialog (QWidget *parent=0)
  Constructor of a CommandsEditDialog object.

- ~CommandsEditDialog ()
  Destructor of CommandEditDialog object.

Private Attributes

- Ui::CommandsEditDialog ui
  The CommandEditDialog, which will be placed in the user interface.

Detailed Description

< Needed to make a dialog.
Constructor & Destructor Documentation

CommandsEditDialog (QWidget *parent = 0)

Parameters:

parent A QWidget (p. 59) representing the parent window.

Precondition:
true

Postcondition:
CommandsEditDialog (p. 15) is set up in the user interface.

~CommandsEditDialog ()

Member Data Documentation

Ui::CommandsEditDialog ui [private]
The documentation for this class was generated from the following files:

- client/commandseditdialog.h
- client/commandseditdialog.cpp

3.1.10 CommunicationLogsDialog Class Reference

The part of the user interface in which communication logs can be viewed.

#include <communicationlogsdialog.h>
Inherits QDialog.

Inheritance diagram for CommunicationLogsDialog:

Public Member Functions

- CommunicationLogsDialog (QWidget *parent=0)
- ~CommunicationLogsDialog ()
  Destructor of CommunicationLogsDialog (p. 16) object.
Private Attributes

- Ui::CommunicationLogsDialog ui

The CommunicationLogsDialog (p. 16), which will be placed in the user interface.

Detailed Description

< Needed to make a dialog.

Constructor & Destructor Documentation

CommunicationLogsDialog (QWidget ∗ parent = 0)
Constructor of a CommunicationLogsDialog (p. 16) object.

Parameters:

parent A QWidget (p. 59) representing the parent window.

Precondition:

ture

Postcondition:

CommunicationLogsDialog (p. 16) is set up in the user interface.

~CommunicationLogsDialog ()

Member Data Documentation

Ui::CommunicationLogsDialog ui [private]
The documentation for this class was generated from the following files:

- client/communicationlogsdialog.h
- client/communicationlogsdialog.cpp

3.1.11 CreateExperimentDialog Class Reference

The part of the user interface which handles the communication logs.

#include <createexperimentdialog.h>

Inherits QDialog.
Inheritance diagram for CreateExperimentDialog:

```
Dialog
   CreateExperimentDialog
   - experimentId
   - initializing
   - ui
   + CreateExperimentDialog()
   + getId()
   + getPayloadId()
   + getSatelliteId()
   + ~CreateExperimentDialog()
   + experimentCreated
   + on_buttonBox_accepted
   + on_satelliteComboBox_currentIndexChanged
```

Public Slots

- void `experimentCreated` (int id)
  Slot which checks if the experiment is created.

- void `on_buttonBox_accepted` ()
  Slot which creates a new experiment.

- void `on_satelliteComboBox_currentIndexChanged` (int index)
  Slot which loads the payloads belonging to a satellite, once a different satellite is chosen.

Public Member Functions

- CreateExperimentDialog (QModelIndex payloadIndex, QWidget *parent=0)
- int `getId` () const
  Gets the identifier of the experiment.

- int `getPayloadId` () const
  Gets the identifier of the payload belonging to the experiment.

- int `getSatelliteId` () const
  Gets the identifier of the satellite belonging to the experiment.

- ~CreateExperimentDialog ()
  Destructor of CreateExperimentDialog (p. 17) object.

Private Attributes

- int `experimentId`
  Identifies the current experiment.
CHAPTER 3. COMPONENT DESCRIPTIONS

- bool initializing
  Indicates whether the client is initializing.

- Ui::CreateExperimentDialog ui
  The CreateExperimentDialog (p. 17), which will be placed in the user interface.

Detailed Description

< Needed to make a dialog. Need to make models.

Constructor & Destructor Documentation

CreateExperimentDialog (QModelIndex payloadIndex, QWidget * parent = 0)
Constructor of a CreateExperimentDialog (p. 17) object.

Parameters:
  parent A QWidget (p. 59) representing the parent window.

Precondition:
  true

Postcondition:
  CreateExperimentDialog (p. 17) is set up in the user interface.

~CreateExperimentDialog ()

Member Function Documentation

void experimentCreated (int id) [slot]
Precondition:
  true

Postcondition:
  the experiment is either accepted or rejected.

int getId () const
Precondition:
  true.

Returns:
  The identifier of the experiment.
CHAPTER 3. COMPONENT DESCRIPTIONS

int getPayloadId () const

Precondition:
true

Returns:
The identifier of the payload belonging to the experiment.

int getSatelliteId () const

Precondition:
true

Returns:
The identifier of the satellite belonging to the experiment.

void on_buttonBox_accepted () [slot]

Precondition:
experimentname <> "" \ (experiment.getPayloadId()).isValid()

Postcondition:
the experiment is created.

void on_satelliteComboBox_currentIndexChanged (int index) [slot]

Precondition:
true

Postcondition:
The payloads of the selected satellite are displayed in the payload combobox.

Member Data Documentation

int experimentId [private]

bool initializing [private]

Ui::CreateExperimentDialog ui [private]

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

- client/createexperimentdialog.h
- client/createexperimentdialog.cpp
3.1.12 Experiment Class Reference

#include <objects.h>

Public Attributes

- int id
- QString name
- QVector<PossibleCommand> possibleCommands
- int scheduledFrom
- int scheduledTo
- QVector<TimedCommand> timedCommands

Detailed Description

Struct representing an experiment, including lists of possible commands and test sequence.

Member Data Documentation

int id

QString name

QVector<PossibleCommand> possibleCommands

int scheduledFrom

int scheduledTo

QVector<TimedCommand> timedCommands

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

- client/objects.h

3.1.13 ExperimentDialog Class Reference

#include <experimentdialog.h>

Inherits QMainWindow.
Inheritance diagram for ExperimentDialog:

```plaintext
QMainWindow

ExperimentDialog
- experimentId
- experimentName
- passesModel
- payloadId
- satelliteId
- scheduledFrom
- sequenceModel

+ ExperimentDialog()
+ ~ExperimentDialog()
+ currentTimedCommandChanged
+ experimentReceived
+ experimentScheduled
+ experimentUnscheduled
+ on_buttonBox_clicked
+ on_commandAddButton_clicked
+ on_commandDeleteButton_clicked
+ on_commandEditButton_clicked
+ on_commandsWithSelectionChanged
+ on_editButton_clicked
+ on_observersButton_clicked
+ on_scheduleButton_clicked
+ on_scheduleDateEdit_dateChanged
+ on_scheduleExperimentButton_clicked
+ on_unscheduleButton_clicked
+ opportunitiesReceived

# closeEvent()
- addTimedCommand()
- setExperimentChanged()
- setupModels()
- switchSchedule()
```

### Public Slots

- `void currentTimedCommandChanged ()`
  
  Slot which enables buttons for a timed command.

- `void experimentReceived (Experiment *experiment)`
  
  Slot which handles an incoming experiment. That is adding possible commands and adding the timed commands to the internal data representations.

- `void experimentScheduled (bool success)`
  
  Slot checks if the experiment is scheduled with success or not.

- `void experimentUnscheduled (bool success)`
  
  Slot which returns whether the experiment was unscheduled.

- `void on_buttonBox_clicked (QAbstractButton *button)`
  
  Slot which updates the experiment in the server.

- `void on_commandAddButton_clicked ()`
  
  Slot which adds a command to the experiment list.

- `void on_commandDeleteButton_clicked ()`

---

22 DDD 0.1
CHAPTER 3. COMPONENT DESCRIPTIONS

Slot which deletes the selected command.

- void on_commandEditButton_clicked ()
  Slot which takes the selected command to the edit modus.

- void on_commands_selectionChanged (bool hasSelection)
  Slot which updates the action when a different commands is selected.

- void on_editButton_clicked ()
  Slot which switches the experiment view to the editing part of that view.

- void on_observersButton_clicked ()
  Slot which opens the manage observers windows.

- void on_scheduleButton_clicked ()
  Slot which switches the experiment view to the scheduling part of that view.

- void on_scheduleDateEdit_dateChanged ()
  Slot which loads the list of oppurtunities for the selected date.

- void on_scheduleExperimentButton_clicked ()
  Slot which schedules the experiment at a given time.

- void on_unscheduleButton_clicked ()
  Slot which unschedules the experiment.

- void opportunitiesReceived (QList< Opportunity > opportunities)
  Slot which processes an incoming list of oppurtunities.

Public Member Functions

- ExperimentDialog (int id, int payloadId, int satelliteId, QWidget *parent=0)
- ~ExperimentDialog ()
  Destructor of CreateExperimentDialog (p. 17) object.

Protected Member Functions

- void closeEvent (QCloseEvent *e)
  Slot which warns the user when he or she wants to close an unsaved experiment.

Private Types

- enum { TypeRole = Qt::UserRole + 1, IntervalFromRole, IntervalToRole, GssIdRole }
- enum { CommandIdRole = Qt::UserRole + 1, ValueRole }
  Enumerator of different roles.
- enum ItemType { Interval, GssId }
Private Member Functions

- void addTimedCommand (const TimedCommand &cmd)
  Add the timed command cmd to the list of experiments.

- void setExperimentChanged (bool changed)
  (Un)sets the save button on the experiment view.

- void setupModels ()
  Sets up all models which are needed in the experiment view.

- void switchSchedule (bool viewSchedule)
  Sets the proper buttons, boxes and other layout items needed for the scheduling view.

Private Attributes

- int experimentId
  Identifies the experiment.

- QString experimentName
  The name of the experiment.

- QStandardItemModel * passesModel
  Needed for the model view controller.

- int payloadId
  Identifies the payload.

- int satelliteId
  Identifies the satellite.

- QDateTime scheduledFrom
- QStandardItemModel * sequenceModel
- Ui::ExperimentDialog ui

  The ExperimentDialog (p. 21), which will be placed in the user interface.

Member Enumeration Documentation

anonymous enum [private]

Enumerator:

TypeRole
IntervalFromRole
IntervalToRole
GssIdRole
anonymous enum  [private]

Enumerator:

   CommandIdRole
   ValueRole

enum ItemType  [private]

Enumerator:

   Interval
   GssId

Constructor & Destructor Documentation

ExperimentDialog (int id, int payloadId, int satelliteId, QWidget ∗ parent = 0)
 Constructor of an ExperimentDialog (p. 21) object.

Parameters:

   id Identifying an experiment.
   payloadId Identifying a payload.
   satelliteId Identifying a satellite.
   parent A QWidget (p. 59) representing the parent window.

Postcondition:

   CreateExperimentDialog (p. 17) is set up in the user interface.

Todo

   Invent some userfriendly way of blocking access until data is loaded.

~ExperimentDialog ()

Member Function Documentation

void addTimedCommand (const TimedCommand & cmd)  [private]

Parameters:

   cmd A timed command.

Precondition:

   Parameter value is valid.

Postcondition:

   The timed command cmd is added to the list of experiments.
Todo

Setting this to true will work but causes some unwanted effects. Implement moving of commands in the list.

Todo

What will happen if !isValid()?

```
void closeEvent (QCloseEvent * e) [protected]
```

Parameters:

- `e` A close event.

Precondition:

The project is unsaved.

Postcondition:

A warning is displayed to the user.

```
void currentTimedCommandChanged () [slot]
```

Postcondition:

If a timedcommand is selected, the edit and delete buttons are enabled, else they are not.

```
void experimentReceived (Experiment * experiment) [slot]
```

Postcondition:

The experiment is loaded in the client application.

```
void experimentScheduled (bool success) [slot]
```

Postcondition:

If the scheduling of an experiment fails, a warning is displayed.

```
void experimentUnscheduled (bool success) [slot]
```

Postcondition:

If the unscheduling failed, a warning will be displayed.
void on_buttonBox_clicked (QAbstractButton * button) [slot]

Parameters:

button The button which is clicked.

Postcondition:

A request to update the experiment is sent to the server.

void on_commandAddButton_clicked () [slot]

Precondition:

A command is selected.

Postcondition:

The selected command is added to the experiment list.

Todo

Disable add button when no command selected.

void on_commandDeleteButton_clicked () [slot]

Precondition:

The sequence view, in which all commands of the experiment are listed, has the focus.

Postcondition:

The selected command is deleted.

void on_commandEditButton_clicked () [slot]

Precondition:

The sequence view, in which all commands of the experiment are listed, has the focus.

Postcondition:

The selected command is taken to the edit modus.

void on_commands_selectionChanged (bool hasSelection) [slot]

void on_editButton_clicked () [slot]

Precondition:

The experiment view is on the scheduling part.

Postcondition:

The experiment view is on the editing part.
void on_observersButton_clicked () [slot]

Postcondition:

The manage observers window is opened.

Todo

Implement ManageObservers (p. 38) dialog.

void on_scheduleButton_clicked () [slot]

Precondition:

The experiment view is on the editing part.

Postcondition:

The experiment view is on the scheduling part.

void on_scheduleDateEdit_dateChanged () [slot]

Postcondition:

The list of opportunities for the selected day is requested.

void on_scheduleExperimentButton_clicked () [slot]

Postcondition:

The schedule is sent to the scheduler in the server.

void on_unscheduleButton_clicked () [slot]

Postcondition:

A request to unschedule the experiment is made on the server.

void opportunitiesReceived (QList< Opportunity > opportunities) [slot]

Postcondition:

The list of opportunities is displayed in the user interface.

void setExperimentChanged (bool changed) [private]

Postcondition:

If the experiment is changed, the save button is enabled. If nothing is changed, the save button is disabled.
void setupModels () [private]

Postcondition:
All models of the experiment view are set up.

void switchSchedule (bool viewSchedule) [private]

Precondition:
The experiment view is on the editing part.

Postcondition:
The experiment view is on the scheduling part.

Member Data Documentation

int experimentId [private]

QString experimentName [private]

QStandardItemModel* passesModel [private]

int payloadId [private]

int satelliteId [private]

QDateTime scheduledFrom [private]

QStandardItemModel* sequenceModel [private]

Ui::ExperimentDialog ui [private]

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

- client/experimentdialog.h
- client/experimentdialog.cpp

3.1.14 LoginDialog Class Reference

Needed for the log in animation.

#include <logindialog.h>

Inherits QDialog.
Inheritance diagram for LoginDialog:

```
<table>
<thead>
<tr>
<th>QDialog</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoginDialog</td>
</tr>
<tr>
<td>- loginAniTimeline</td>
</tr>
<tr>
<td>- ui</td>
</tr>
<tr>
<td>- noPasswordChange</td>
</tr>
<tr>
<td>+ LoginDialog()</td>
</tr>
<tr>
<td>+ ~LoginDialog()</td>
</tr>
<tr>
<td>+ on_cancelButton_clicked</td>
</tr>
<tr>
<td>+ on_loginButton_clicked</td>
</tr>
<tr>
<td>+ on_preferencesButton_clicked</td>
</tr>
<tr>
<td>+ reject</td>
</tr>
<tr>
<td># closeEvent()</td>
</tr>
<tr>
<td>- requestFinished</td>
</tr>
<tr>
<td>- setAniFrame</td>
</tr>
</tbody>
</table>
```

Public Slots

- void `on_cancelButton_clicked()`
  
  Slot which cancels the logging in.

- void `on_loginButton_clicked()`
  
  Slot which handles the login of the client.

- void `on_preferencesButton_clicked()`
  
  Slot which invokes the preferences dialog.

- virtual void `reject()`
  
  Slot which handles the closing of the client.

Public Member Functions

- `LoginDialog (QWidget *parent=0, Qt::WindowFlags f=0)`
  
  Destructor of `LoginDialog` (p. 29) object.

Protected Member Functions

- virtual void `closeEvent (QCloseEvent *e)`
  
  Slot which warns the user when he or she wants to close the client.

Private Slots

- void `requestFinished ()`
CHAPTER 3. COMPONENT DESCRIPTIONS

Slot which handles the result of the request.

- void setAniFrame (int frame)
  Slot which sets the frame of the login animation to a specific frame number.

Private Attributes

- QTImeline * loginAniTimeLine
- Ui::LoginDialog ui
  The LoginDialog (p. 29), which will be placed in the user interface.

Static Private Attributes

- static const QString noPasswordChange
  Used to hide the password on the screen.

Constructor & Destructor Documentation

LoginDialog (QWidget * parent = 0, Qt::WindowFlags f = 0)
Constructor of an LoginDialog (p. 29) object.

Parameters:
  parent A QWidget (p. 59) representing the parent window.
  f Windowflags which are parameters of the dialog.

Precondition:
  true

Postcondition:
  LoginDialog (p. 29) is set up in the user interface.

∼LoginDialog ()

Member Function Documentation

void closeEvent (QCloseEvent * e) [protected, virtual]
Parameters:
  e A close event.

Precondition:
  true

Postcondition:
  A warning is displayed to the user.
void on_cancelButton_clicked () [slot]

Precondition:
true

Postcondition:
The logging in is cancelled.

void on_loginButton_clicked () [slot]

Precondition:
true

Postcondition:
The client tries to login to a server.

Todo
Make a save password checkbox (now always saving password).

void on_preferencesButton_clicked () [slot]

Precondition:
true

Postcondition:
The preferences dialog is shown on screen.

void reject () [virtual, slot]

Precondition:
true

Postcondition:
The client is closed.

void requestFinished () [private, slot]

Precondition:
true

Postcondition:
When the client is logged in, nothing is printed, if there occurred an error, that error is printed.
void setAniFrame (int frame) [private, slot]

Parameters:

frame The frame to which the animation will be set.

Precondition:

true

Postcondition:

A new frame is displayed on screen when needed.

Member Data Documentation

QTimeLine* loginAniTimeLine [private]
A timeline for the log in animation.

const QString noPasswordChange [static, private]

Ui::LoginDialog ui [private]

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

- client/logindialog.h
- client/logindialog.cpp

3.1.15 MainWindow Class Reference

Needed for the model view controller.

#include <mainwindow.h>

Inherits QMainWindow.
CHAPTER 3. COMPONENT DESCRIPTIONS

Inheritance diagram for MainWindow:

```
QMainWindow
- MainWindow()
- ~MainWindow()
- experimentListUpdated
- on_actionAbout_triggered
- on_actionCommunicationLogs_triggered
- on_actionEditExperiment_triggered
- on_actionLogout_triggered
- on_actionManageObservers_triggered
- on_actionManageSatellites_triggered
- on_actionManageUsers_triggered
- on_actionNewExperiment_triggered
- on_actionObservationData_triggered
- on_actionObserveExperiment_triggered
- on_actionPreferences_triggered
- on_actionRefreshList_triggered
- on_actionScheduleExperiment_triggered
- on_experimentsTreeView_doubleClicked
- selectionChanged
  $ closeEvent()
- closeOtherWindows()
- getSelectedExperiment()
- getSelectedExperiments()
- openAdmin()
- setupModels()
```

Public Slots

- void `experimentListUpdated()`
  
  Slot which is a callback for the receiving of a new list of experiments. This slot updates the view, expanding all entries and resizing the columns, if necessary.

- void `on_actionAbout_triggered()`
  
  Slot which shows the about window.

- void `on_actionAboutQt_triggered()`
  
  Slot which shows the about Qt window.

- void `on_actionCommunicationLogs_triggered()`
  
  Slot which shows the logs of communications.

- void `on_actionEditExperiment_triggered()`
  
  Slot which shows the edit experiment dialog in editing mode.

- void `on_actionLogout_triggered()`
  
  Slot which handles the logging out of the client.

- void `on_actionManageObservers_triggered()`
  
  Slot which opens the manage observers dialog.
• void on_actionManageSatellites_triggered ()
  Slot which shows the manage satellites dialog.

• void on_actionManageUsers_triggered ()
  Slot which shows the manage users dialog.

• void on_actionNewExperiment_triggered ()
  Slot which handles the showing of the new experiment dialog.

• void on_actionObservationData_triggered ()
  Slot which shows the observation history of an experiment.

• void on_actionObserveExperiment_triggered ()
  Slot which shows the observation dialog.

• void on_actionPreferences_triggered ()
  Slot which handles the showing of the preferences dialog.

• void on_actionRefreshList_triggered ()
  Slot which refreshes the main view.

• void on_actionScheduleExperiment_triggered ()
  Slot which shows the edit experiment dialog in scheduling mode.

• void on_experimentsTreeView_doubleClicked (const QModelIndex &index)
  This slot handles double clicking of experiments in the main window. This opens the edit experiment dialog.

• void selectionChanged ()
  This slot activates and deactivates the appropriate options for each item on the main window. In particular, only experiments which are scheduled or local can be edited or scheduled.

Public Member Functions

• MainWindow (QWidget *parent=0)

• ~MainWindow ()
  Destructor of MainWindow (p. 33) object.

Protected Member Functions

• virtual void closeEvent (QCloseEvent *e)
  Gracefully closes the client, shutting down the connection with the server.
CHAPTER 3. COMPONENT DESCRIPTIONS

Private Member Functions

- **bool closeOtherWindows ()**
  Closes all windows except the main window and the login dialog.

- **QModelIndex getSelectedExperiment ()**
  Gets the first experiment selected by the user.

- **QList<int> getSelectedExperiments ()**

- **void openAdmin (QString function, QString extra=QString())**
  Opens the administration front end in the user’s default browser. The user is automatically logged in to the front end.

- **void setupModels ()**
  Requests the server connectivity to update the list of experiments.

Private Attributes

- **int closing**
  Keeps current state in shutdown procedure.

- **bool dontLogout**
  Prevents sending logout signal.

- **Ui::MainWindow ui**

Constructor & Destructor Documentation

MainWindow (QWidget * parent = 0)
Constructor of an MainWindow (p. 33) object.

Parameters:

  parent A QWidget (p. 59) representing the parent window.

Postcondition:

MainWindow (p. 33) is set up in the user interface.

~MainWindow ()

Member Function Documentation

void closeEvent (QCloseEvent * e) [protected, virtual]

Todo

Show messagebox if QApplication::topLevelWidgets().count() > 1?
bool closeOtherWindows () [private]

void experimentListUpdated () [slot]

QModelIndex getSelectedExperiment () [private]

QList<int> getSelectedExperiments () [private]

void on_actionAbout_triggered () [slot]

void on_actionAboutQt_triggered () [slot]

void on_actionCommunicationLogs_triggered () [slot]

void on_actionEditExperiment_triggered () [slot]

void on_actionLogout_triggered () [slot]

Precondition:
    All other windows are closed \ dontLogout == false.

Postcondition:
    The client is closed and the user is logged out with the server.

void on_actionManageObservers_triggered () [slot]

void on_actionManageSatellites_triggered () [slot]

void on_actionManageUsers_triggered () [slot]

void on_actionNewExperiment_triggered () [slot]

void on_actionObservationData_triggered () [slot]

void on_actionObserveExperiment_triggered () [slot]

void on_actionPreferences_triggered () [slot]

void on_actionRefreshList_triggered () [slot]

void on_actionScheduleExperiment_triggered () [slot]
void on_experimentsTreeView_doubleClicked (const QModelIndex & index) [slot]

void openAdmin (QString function, QString extra = QString()) [private]

void selectionChanged () [slot]

Todo
Not yet implemented: delete experiment function.

Todo
Not yet implemented: manage observers for an experiment.

void setupModels () [private]
Requests the server connectivity to update the list of experiments after this list is updated, it is refreshed on the window. Furthermore, several parts of the GUI are connected to the new list of experiments.

Member Data Documentation

int closing [private]

bool dontLogout [private]

Ui::MainWindow ui [private]
The documentation for this class was generated from the following files:

- client/mainwindow.h
- client/mainwindow.cpp

3.1.16 ManageObservers Class Reference

#include <manageobservers.h>
Inherits QDialog.
CHAPTER 3. COMPONENT DESCRIPTIONS

Inheritance diagram for ManageObservers:

```
QDialog

ManageObservers
- observerModel
- ui
- userModel
+ ManageObservers()
+ ~ManageObservers()
- addUser()
- setupModels()
```

Public Member Functions

- `ManageObservers (QWidget *parent=0)`
- `~ManageObservers ()`

Private Member Functions

- `void addUser (const QString &first, const QString &last)`
- `void setupModels ()`

Private Attributes

- ` UserModel * observerModel`
- `Ui::ManageObserversDialog ui`
- ` UserModel * userModel`

Constructor & Destructor Documentation

ManageObservers (QWidget * parent = 0)

Todo

Check for duplicates when copying users.

`~ManageObservers ()`

Member Function Documentation

`void addUser (const QString & first, const QString & last) [private]`

`void setupModels () [private]`
Member Data Documentation

UserModel * observerModel [private]

Ui::ManageObserversDialog ui [private]

UserModel * userModel [private]

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

- client/manageobservers.h
- client/manageobservers.cpp

3.1.17 ManagePayloadsDialog Class Reference

#include <managepayloadsdialog.h>

Inherits QDialog.

Inheritance diagram for ManagePayloadsDialog:

![Inheritance Diagram]

Public Slots

- void on_addButton_clicked ()
- void on_editButton_clicked ()

Public Member Functions

- ManagePayloadsDialog (QWidget *parent=0)
- ~ManagePayloadsDialog ()

Private Attributes

- Ui::ManagePayloadsDialog ui

Constructor & Destructor Documentation

ManagePayloadsDialog (QWidget * parent = 0)
CHAPTER 3. COMPONENT DESCRIPTIONS

~ManagePayloadsDialog ()

Member Function Documentation

void on_addButton_clicked () [slot]

void on_editButton_clicked () [slot]

Member Data Documentation

Ui::ManagePayloadsDialog ui [private]
The documentation for this class was generated from the following files:

- client/managepayloadsdialog.h
- client/managepayloadsdialog.cpp

3.1.18 ManageSatellitesDialog Class Reference

#include <managesatellitesdialog.h>
Inherits QDialog.

Inheritance diagram for ManageSatellitesDialog:

Public Slots

- void on_addButton_clicked ()
- void on_editButton_clicked ()

Public Member Functions

- ManageSatellitesDialog (QWidget *parent=0)
- ~ManageSatellitesDialog ()

Private Attributes

- Ui::ManageSatellitesDialog ui
CHAPTER 3. COMPONENT DESCRIPTIONS

Constructor & Destructor Documentation

ManageSatellitesDialog (QWidget * parent = 0)

~ManageSatellitesDialog ()

Member Function Documentation

void on_addButton_clicked () [slot]

void on_editButton_clicked () [slot]

Member Data Documentation

Ui::ManageSatellitesDialog ui [private]

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

- client/managesatellitesdialog.h
- client/managesatellitesdialog.cpp

3.1.19 ManageUsersDialog Class Reference

#include <manageusersdialog.h>

Inherits QDialog.

Inheritance diagram for ManageUsersDialog:

QDialog

ManageUsersDialog

- ui
+ ManageUsersDialog
+ ~ManageUsersDialog
+ on_addButton_clicked
+ on_editButton_clicked

Public Slots

- void on_addButton_clicked ()
- void on_editButton_clicked ()

Public Member Functions

- ManageUsersDialog (QWidget *parent=0)
- ~ManageUsersDialog ()
CHAPTER 3. COMPONENT DESCRIPTIONS

Private Attributes

- Ui::ManageUsersDialog ui

Constructor & Destructor Documentation

ManageUsersDialog (QWidget ∗ parent = 0)

~ManageUsersDialog ()

Member Function Documentation

void on_addButton_clicked () [slot]

void on_editButton_clicked () [slot]

Member Data Documentation

Ui::ManageUsersDialog ui [private]

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

- client/manageusersdialog.h
- client/manageusersdialog.cpp

3.1.20 MultiHttp Class Reference

#include <multihttp.h>

InheritsQObject.

Inheritance diagram for MultiHttp:

```
QObject

MultiHttp

- https
- requestTypes
- responses
- abort()
- MultiHttp()
- request()
- requestPending
- setHost()
- setProxy()
- ~MultiHttp()
- requestFinishedSlot
```
CHAPTER 3. COMPONENT DESCRIPTIONS

Signals

- void requestFinished (int type, QByteArray *response, bool error)

Public Member Functions

- void abort (int slot=-1)
- MultiHttp (int concurrency, QObject *parent=0)
- void request (const QHttpRequestHeader &header, const QByteArray &data, int type, int slot)
- bool requestPending (int slot, int type)
- void setHost (const QString &hostName, quint16 port=80)
- void setProxy (const QString &host, int port, const QString &username=QString(), const QString &password=QString())
- ~MultiHttp ()

Private Slots

- void requestFinishedSlot (int id, bool error)

Private Attributes

- QVector<QHttp *> https
- QVector< QMap<int, int> > requestTypes
- QVector<QByteArray *> responses

Detailed Description

A wrapper for multiple QHttp’s, thus implementing concurrency.

Constructor & Destructor Documentation

MultiHttp (int concurrency, QObject *parent = 0)

Parameters:

concurrency Number of concurrent connections.

parent Parent QObject (p. 57).

~MultiHttp ()

Member Function Documentation

void abort (int slot = -1)
Abort QHttp in slot slot. All QHttp’s are aborted if -1 is used.
CHAPTER 3. COMPONENT DESCRIPTIONS

void request (const QHttpRequestHeader & header, const QByteArray & data, int type, int slot)
Send a request to QHttp in slot slot. Remember type of request.

void requestFinished (int type, QByteArray * response, bool error) [signal]
Emitted when a request is finished. Should be connected using Qt::QThreadConnection because some cleaning up is done afterwards.

Parameters:
  type  The type given when the request was made.
  response  Response from server.
  error  From QHttp.

void requestFinishedSlot (int id, bool error) [private, slot]

bool requestPending (int slot, int type)
Tells whether a request of type type is pending in slot slot.

void setHost (const QString & hostName, quint16 port = 80)
Set host for all QHttp objects. See QHttp documentation.

void setProxy (const QString & host, int port, const QString & username = QString(),
const QString & password = QString())
Set proxy for all QHttp objects. See QHttp documentation.

Member Data Documentation
QVector<QHttp *> https [private]

QVector<QMap<int, int>> requestTypes [private]

QVector<QByteArray *> responses [private]
The documentation for this class was generated from the following files:
  • client/multihttp.h
  • client/multihttp.cpp

3.1.21 ObservationDataDialog Class Reference

#include <observationdatadialog.h>
Inherits QDialog.

DDD 0.1 45
CHAPTER 3. COMPONENT DESCRIPTIONS

Inheritance diagram for ObservationDataDialog:

```
QDialog

ObservationDataDialog
- dataModel
- initializing
- ui
- thumbnailSize
+ ObservationDataDialog()
+ ~ObservationDataDialog()
- newThumbnail
- on_filterButton_clicked
- on_payloadComboBox_currentIndexChanged
- on_satelliteComboBox_currentIndexChanged
```

Public Member Functions

- **ObservationDataDialog** (QModelIndex index, QWidget *parent=0)
  
  *Constructs the observation data dialog.*

- **~ObservationDataDialog ()**
  
  *Destroys the observation data dialog.*

Private Slots

- void **newThumbnail** (Thumbnail *thumbnail)
  
  *Shows an arrived thumbnail.*

- void **on_filterButton_clicked ()**
  
  *Passes the filter to the server.*

- void **on_payloadComboBox_currentIndexChanged** (int index)
  
  *Changes the payload for which experiments can be chosen.*

- void **on_satelliteComboBox_currentIndexChanged** (int index)
  
  *Changes the satellite for which payloads and experiments can be chosen.*

Private Attributes

- QStandardItemModel * dataModel
- bool initializing
- Ui::ObservationDataDialog ui

Static Private Attributes

- static const int thumbnailSize = 128
Constructor & Destructor Documentation

ObservationDataDialog (QModelIndex index, QWidget * parent = 0)

~ObservationDataDialog ()

Member Function Documentation

void newThumbnail (Thumbnail * thumbnail) [private, slot]

Parameters:

  thumbnail Pointer to the thumbnail to be shown.

void on_filterButton_clicked () [private, slot]

void on_payloadComboBox_currentIndexChanged (int index) [private, slot]

void on_satelliteComboBox_currentIndexChanged (int index) [private, slot]

Member Data Documentation

QStandardItemModel* dataModel [private]

bool initializing [private]

const int thumbnailSize = 128 [static, private]

Ui::ObservationDataDialog ui [private]

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

• client/observationdatadialog.h
• client/observationdatadialog.cpp

3.1.22 ObservationView Class Reference

#include <observationview.h>
Inherits QWidget.
CHAPTER 3. COMPONENT DESCRIPTIONS

Inheritance diagram for ObservationView:

![Inheritance Diagram]

### Public Types

- `enum Orientation { Left, Right }`

### Public Slots

- `void appendLog (QString message)`
  
  Append message to the log view.

- `void newObservation (QVariant observation)`
  
  Show observation in the interface. This can either be a string or a QPixmap.

- `void on_addButton_clicked ()`

### Public Member Functions

- `bool logVisible () const`
  
  Return whether the log view is visible.

- `ObservationView (QWidget *parent=0)`

- `Orientation orientation () const`
  
  Returns the current orientation.

- `bool postProcVisible () const`
  
  Return whether the post processor list is visible.

- `void setLogVisible (bool visible)`
  
  Hide or show the log view.

- `void setOrientation (Orientation orientation)`
CHAPTER 3. COMPONENT DESCRIPTIONS

Sets the orientation for this **ObservationView** (p. 47).

- void `setPostProcVisible` (bool visible)
  
  *Hide or show the post processor list.*

- void `setTitle` (QString title)
  
  *Set title of groupbox.*

- ~`ObservationView` ()

**Properties**

- bool `logVisible`
- `Orientation orientation`
- bool `postProcVisible`

**Private Attributes**

- `Ui::ObservationView ui`

**Detailed Description**

Widget for displaying observations and post processing settings. Also has a log view in the bottom.

**Member Enumeration Documentation**

enum `Orientation`

**Enumerator:**

- `Left`
- `Right`

**Constructor & Destructor Documentation**

`ObservationView` (QWidget ∗ `parent` = 0)

~`ObservationView` ()

**Member Function Documentation**

void `appendLog` (QString `message`) [slot]

bool `logVisible` () const

void `newObservation` (QVariant `observation`) [slot]
void on_addButton_clicked () [slot]

Orientation orientation () const

See also:
  Orientation (p. 49)

bool postProcVisible () const

void setLogVisible (bool visible)

void setOrientation (Orientation orientation)

See also:
  Orientation (p. 49)

void setPostProcVisible (bool visible)

void setTitle (QString title)

Member Data Documentation
Ui::ObservationView ui [private]

Property Documentation
bool logVisible [read, write]

ObservationView::Orientation orientation [read, write]

bool postProcVisible [read, write]

The documentation for this class was generated from the following files:
  * client/observationview.h
  * client/observationview.cpp

3.1.23 Observe Class Reference

Provides functionality for observing an experiment to the client.
#include <observe.h>
Inherits QMainWindow.
Inheritance diagram for Observe:

```
QMainWindow
  Observe
    - experiment1Id
    - experiment2Id
    - ui
    + Observe()
    + ~Observe()
    + newThumbnail
    # closeEvent()
```

**Public Slots**

- void `newThumbnail` (Thumbnail *thumbnail)
  
  Shows a newly arrived thumbnail, replacing the one currently displayed.

**Public Member Functions**

- `Observe` (QModelIndex experiment1, QModelIndex experiment2, QWidget *parent=0)
  
  Constructs the observation dialog.

- `~Observe`
  
  Default destructor.

**Protected Member Functions**

- void `closeEvent` (QCloseEvent *e)

**Private Attributes**

- int `experiment1Id`
- int `experiment2Id`
- Ui::ObserveDialog `ui`

**Constructor & Destructor Documentation**

Observe (QModelIndex `experiment1`, QModelIndex `experiment2`, QWidget `*parent` = 0)

**Parameters:**

- `experiment1` The first experiment to be observed.
**experiment2** The second experiment to be observed.

**Todo**

Set name of groundstation or other info.

**Todo**

Set name of groundstation or other info.

~Observe ()

Member Function Documentation

void closeEvent (QCloseEvent * e) [protected]

void newThumbnail (Thumbnail * thumbnail) [slot]

**Todo**

Make waiting time settable via slider in UI? This would allow a scientist to slow down the image flow to look more carefully at the experiment results.

Member Data Documentation

int experiment1Id [private]

int experiment2Id [private]

Ui::ObserveDialog ui [private]

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

- client/observe.h
- client/observe.cpp

3.1.24 Opportunity Class Reference

#include <objects.h>

Public Attributes

- int end
- QString gssid
- QString remarks
- int start
- bool twoway
CHAPTER 3. COMPONENT DESCRIPTIONS

Detailed Description

Struct representing a booking opportunity.

Member Data Documentation

int end

QString gssid

QString remarks

int start

bool twoway

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

- client/objects.h

3.1.25 PossibleCommand Class Reference

#include <objects.h>

Public Attributes

- int id
- QString name
- QVariant paramDefault
- QVariant paramMax
- QVariant paramMin

Detailed Description

Struct representing a possible command (one that is not included in a test sequence).

Member Data Documentation

int id

QString name

QVariant paramDefault

QVariant paramMax
QVariant paramMin

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

- client/objects.h

3.1.26 Preferences Class Reference

#include <preferences.h>

Inherits QDialog.

Inheritance diagram for Preferences:

```
QDialog
    Preferences
      - ui
      - minPasswordLength
      + Preferences()
      + ~Preferences()
      + on_buttonBox_accepted
      + on_buttonBox_clicked
      + on_buttonBox_rejected
      + on_changePasswordButton_clicked
      + on_newPasswordEdit_textChanged
      + on_retypedNewPasswordEdit_textChanged
      + passwordChanged
      - apply()
```

Public Slots

- void on_buttonBox_accepted ()
- void on_buttonBox_clicked (QAbstractButton *button)
- void on_buttonBox_rejected ()
- void on_changePasswordButton_clicked ()
- void on_newPasswordEdit_textChanged ()
- void on_retypedNewPasswordEdit_textChanged ()
- void passwordChanged (bool ok)

Public Member Functions

- Preferences (QWidget *parent=0)
  
  Default constructor.

- ~Preferences ()
  
  Default destructor.

Private Member Functions

- bool apply ()
Private Attributes

- Ui::PreferenceDialog ui

Static Private Attributes

- static const int minPasswordLength = 6

Constructor & Destructor Documentation

Preferences (QWidget * parent = 0)

~Preferences ()

Member Function Documentation

bool apply () [private]

void on_buttonBox_accepted () [slot]

void on_buttonBox_clicked (QAbstractButton * button) [slot]

void on_buttonBox_rejected () [slot]

void on_changePasswordButton_clicked () [slot]

void on_newPasswordEdit_textChanged () [slot]

void on_retypedNewPasswordEdit_textChanged () [slot]

void passwordChanged (bool ok) [slot]

Called by ServerConnectivity (p. 61) when password change request returned.

Member Data Documentation

const int minPasswordLength = 6 [static, private]

Todo

Fix behaviour when enter is pressed in password boxes.

Ui::PreferenceDialog ui [private]

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

- client/preferences.h
- client/preferences.cpp
3.1.27 QDialog Class Reference

Inherited by CommandsEditDialog, CommunicationLogsDialog, CreateExperimentDialog, LoginDialog, ManageObservers, ManagePayloadsDialog, ManageSatellitesDialog, ManageUsersDialog, ObservationDataDialog, Preferences, and SelectObserveDialog.

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

- client/manageusersdialog.h

3.1.28 QMainWindow Class Reference

Inherited by ExperimentDialog, MainWindow, and Observe.

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

- client/experimentdialog.h
3.1.29 QObject Class Reference

Inherited by MultiHttp, and ServerConnectivity.
CHAPTER 3. COMPONENT DESCRIPTIONS

Inheritance diagram for QObject:

The documentation for this class was generated from the following file:
3.1.30 QStandardItemModel Class Reference

Inherited by UserModel.

Inheritance diagram for QStandardItemModel:

```
QStandardItemModel
  UserModel
    + UserModel()
    + ~UserModel()
```

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

- client/usermodel.h

3.1.31 QWidget Class Reference

Inherited by Commands, ObservationView, and UserSelectionWidget.

Inheritance diagram for QWidget:

```
QWidget
  Commands
    - commandModel
    - parameterModel
      - ui
        + addCommand()
        + clearSelection()
        + Command()
        + getCommand()
        + getCommandName()
        + selectCommand()
        + Commands()
        + addParameter()
        + setupCommands()
        + setupParameterHeader()
        + setupParameters()
        + commandList_currentChanged
  ObservationView
    - ui
      + logVisible()
      + ObservationView
        + orientation()
        + postProcVisible
        + setLogVisible()
        + setOrientation()
        + setTitle()
      + ObservationView()
      + on_addButton_clicked
      + on_removeButton_clicked
      + on_searchUserEdit_textChanged
  UserSelectionWidget
    - fullsetModel
    - fullsetProxyModel
      - SubsetModel
    - subsetModel
      - ui
        + setFullsetModel()
        + setFullsetProxyModel
        + setSubsetModel()
        + setSubsetModel()
        + UserSelectionWidget()
        + ~UserSelectionWidget()
        + on_addButton_clicked
        + on_removeButton_clicked
        + on_searchUserEdit_textChanged
```

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

- client/userselectionwidget.h
CHAPTER 3. COMPONENT DESCRIPTIONS

3.1.32 SelectObserveDialog Class Reference

#include <selectobservedialog.h>

Inherits QDialog.

Inheritance diagram for SelectObserveDialog:

```
QDialog
- initializing
- ui
+ experiment1()
+ experiment2()
+ SelectObserveDialog()
+ ~SelectObserveDialog()
- on_experiment1GroupBox_toggled
- on_experiment2GroupBox_toggled
- on_payload1ComboBox_currentIndexChanged
- on_payload2ComboBox_currentIndexChanged
- on_satellite1ComboBox_currentIndexChanged
- on_satellite2ComboBox_currentIndexChanged
```

Public Member Functions

- QModelIndex experiment1 ()
  Return model index for the first selected experiment.

- QModelIndex experiment2 ()
  Return model index for the second selected experiment.

- SelectObserveDialog (QModelIndex index1, QModelIndex index2, QWidget *parent=0)

- ~SelectObserveDialog ()

Private Slots

- void on_experiment1GroupBox_toggled (bool on)
- void on_experiment2GroupBox_toggled (bool on)
- void on_payload1ComboBox_currentIndexChanged (int index)
- void on_payload2ComboBox_currentIndexChanged (int index)
- void on_satellite1ComboBox_currentIndexChanged (int index)
- void on_satellite2ComboBox_currentIndexChanged (int index)

Private Attributes

- bool initializing
- Ui::SelectObserveDialog ui

Detailed Description

Dialog asking user to select one or two experiments to observe.
CHAPTER 3. COMPONENT DESCRIPTIONS

Constructor & Destructor Documentation

SelectObserveDialog (QModelIndex index1, QModelIndex index2, QWidget * parent = 0)

Construct the dialog and set the initial experiments to index1 and index2. Both may be invalid. If so, the corresponding group boxes will be unchecked.

~SelectObserveDialog ()

Member Function Documentation

QModelIndex experiment1 ()

QModelIndex experiment2 ()

void on_experiment1GroupBox_toggled (bool on) [private, slot]

void on_experiment2GroupBox_toggled (bool on) [private, slot]

void on_payload1ComboBox_currentIndexChanged (int index) [private, slot]

void on_payload2ComboBox_currentIndexChanged (int index) [private, slot]

void on_satellite1ComboBox_currentIndexChanged (int index) [private, slot]

void on_satellite2ComboBox_currentIndexChanged (int index) [private, slot]

Member Data Documentation

bool initializing [private]

Ui::SelectObserveDialog ui [private]

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

- client/selectobservedialog.h
- client/selectobservedialog.cpp

3.1.33 ServerConnectivity Class Reference

#include <serverconnectivity.h>

Inherits QObject.
Inheritance diagram for ServerConnectivity:

```
QObject

ServerConnectivity
- adminUrl_
- challenge
- errorMessage
- experimentListModel
- hashedNewPassword
- hashedPassword_
- loggedIN
- multiHttpRequest
- proxyHost_
- proxyPort_
- serverHost_
- serverPort_
- sessionId
- username_
- usertype_
- instance_

+ adminUrl()
+ changePassword()
+ createExperiment()
+ getAllThumbnails()
+ getExperimentDetails()
+ getExperimentListMode
+ getListOfOpportunities
+ getThumbnail()
+ hashedPassword()
+ isLoggedIn()
+ lastErrorMessage()
+ login()
+ logout()
+ proxyHost()
+ proxyPort()
+ resetState()
+ serverHost()
+ serverPort()
+ setAdminUrl()
+ setPassword()
+ setProxy()
+ setServer()
+ setUsername()
+ stop()
+ unscheduleExperiment()
+ updateExperiment()
+ updateExperimentList
+ username()
+ usertype()
+ instance()
- hash()
- interpretBooleanResponse
- interpretChallengeResponse
- interpretIntResponse
- interpretListOfOpportunities
- interpretLoginResponse
- interpretOpportunity
- interpretThumbnail()
- interpretThumbnails()
- makeHttpHeader()
- processDetailedExperiment
- processExperimentsList
- processPayload()
- processPossibleCommand
- processSatellite
- processTimedCommand()
- readBoolean()
- readByteArray()
- readInt()
- readString()
- readVariant()
- ServerConnectivity()
- simpleRequest()
- ~ServerConnectivity()
- requestFinished
```
CHAPTER 3. COMPONENT DESCRIPTIONS

Public Types

- enum { TypeRole = Qt::UserRole + 1, IdRole }
  Roles for data in experimentListModel.

- enum ItemType { UnknownItemType = 0, ExperimentItemType, PayloadItemType, SatelliteItemType }
  Values for data with TypeRole in experimentListModel.

Signals

- void experimentCreated (int id)
- void experimentListUpdated ()
- void experimentReceived (Experiment *experiment)
  Assumes blocking connection for deleting pointer.

- void experimentScheduled (bool ok)
- void experimentUnscheduled (bool ok)
- void loginFinished ()
- void logoutFinished ()
- void newSingleThumbnail (Thumbnail *thumbnail)
  Assumes blocking connection for deleting pointer.

- void newThumbnail (Thumbnail *thumbnail)
  Assumes blocking connection for deleting pointer.

- void opportunitiesReceived (QList< Opportunity > opportunities)
- void passwordChanged (bool ok)

Public Member Functions

- QString adminUrl () const
  Returns the URL of the administration frontend.

- void changePassword (QString oldPassword, QString newPassword)
- void createExperiment (QString name, int payloadId)
- void getAllThumbnails (int experimentId)
- void getExperimentDetails (int id)
- QAbstractItemModel * getExperimentListModel () const
- void getListOfOpportunities (int experimentID, int from, int to)
- void getThumbnail (int experimentId)
- QByteArray hashedPassword () const
  Returns the SHA-256 hashed password of the user logged in.

- bool isLoggedIn () const
- QString lastErrorMessage () const
- void login ()
  Requests to log in username with hashedPassword.
CHAPTER 3. COMPONENT DESCRIPTIONS

- **void logout ()**
  
  Requests to log out username. Reply in logoutFinished() (p. 70).

- **QString proxyHost () const**
  
  Returns the hostname of the proxy.

- **quint16 proxyPort () const**
  
  Returns the port the proxy is listening on.

- **void resetState ()**
  
  Resets the state of the client.

- **void scheduleExperiment (int experimentID, int startTime, QList<Opportunity> opps)**

- **QString serverHost () const**
  
  Returns the server’s hostname.

- **quint16 serverPort () const**
  
  Returns the port the server is listening on.

- **void setAdminUrl (QString url)**
  
  Sets the URL of the administration frontend.

- **void setPassword (QString password)**
  
  Sets the SHA-256 hash of the password.

- **void setProxy (QString host, quint16 port)**
  
  Sets the hostname and port of the proxy.

- **void setServer (QString host, quint16 port)**
  
  Sets the server information.

- **void setUsername (QString username)**
  
  Sets the username.

- **void stop ()**
  
  Stops the connectivity and deletes it.

- **void unscheduleExperiment (int experimentID)**

- **void updateExperiment (Experiment *experiment)**

- **void updateExperimentList ()**
  
  Updates the experiment list maintained in experimentListModel.

- **QString username () const**
  
  Returns the name of the user logged in.

- **QString usertype () const**
CHAPTER 3. COMPONENT DESCRIPTIONS

Static Public Member Functions

- static ServerConnectivity * instance ()

  Constructs an instance of the connectivity, if it not yet exists, and returns the instance.

Private Types

- enum HttpSlot { GeneralSlot = 0, MccSlot = 1, ObserveSlot = 2 }
- enum RequestType { ChallengeRequest = 1, LoginRequest, LogoutRequest, ExperimentListRequest, ExperimentRequest, PasswordChangeRequest, CreateExperimentRequest, UpdateExperimentRequest, GetListOfOpportunitiesRequest, ScheduleExperimentRequest, ThumbnailRequest, AllThumbnailsRequest, UnscheduleExperimentRequest }

Private Slots

- void requestFinished (int type, QByteArray *response, bool error)

Private Member Functions

- QByteArray hash (QByteArray data)
- bool interpretBooleanResponse (QByteArray *response, bool *result)
- bool interpretChallengeResponse (QByteArray *response)
- bool interpretIntResponse (QByteArray *response, int *result)
- bool interpretListOfOpportunities (QByteArray *response, QList< Opportunity > *opportunities)
- bool interpretLoginResponse (QByteArray *response, bool *result)
- bool interpretOpportunity (QDomElement *e, Opportunity *opportunity)
- bool interpretThumbnail (QByteArray *response, Thumbnail *thumbnail)
- bool interpretThumbnails (QByteArray *response)
- QHttpRequestHeader makeHttpHeader ()
- bool processDetailedExperiment (QByteArray *response, Experiment *experiment)
- void processExperiment (QDomElement *e, QStandardItem *payload)
- void processExperimentsList (QByteArray *response)
- void processPossibleCommand (QDomElement *e, PossibleCommand *cmd)
- void processSatellite (QDomElement *e)
- void processTimedCommand (QDomElement *e, TimedCommand *cmd)
- bool readBoolean (QDomElement *e, bool *result)
- bool readByteArray (QDomElement *e, QByteArray *result)
- bool readInt (QDomElement *e, int *result)
- bool readString (QDomElement *e, QString *result)
- bool readVariant (QDomElement *e, QVariant *result)
- ServerConnectivity ()
- void simpleRequest (int type, QString methodName, QString param=QString())
- ~ServerConnectivity ()
CHAPTER 3. COMPONENT DESCRIPTIONS

Private Attributes

- QString adminUrl_
- QByteArray challenge
- QString errorMessage
- QStandardItemModel * experimentListModel
- QByteArray hashedNewPassword
- QByteArray hashedPassword_
- bool isLoggedIn
- MultiHttp multiHttp
- QString proxyHost_
- quint16 proxyPort_
- QString serverHost_
- quint16 serverPort_
- QByteArray sessionId
- QString username_
- QString usertype_

Static Private Attributes

- static ServerConnectivity * instance_ = 0

Detailed Description

This class handles communication with the server. Specific instances of this class are retrieved as per singleton pattern.

Member Enumeration Documentation

anonymous enum

Enumerator:

  TypeRole
  IdRole

enum HttpSlot [private]

Numbers for HTTP slots.

Enumerator:

  GeneralSlot
  MccSlot  For non-blocking requests.
  ObserveSlot  For requests that need communication with MCC.

enum ItemType

Enumerator:

UnknownItemType
ExperimentItemType
PayloadItemType
SatelliteItemType

enum RequestType [private]

Request types used to identify responses.

Enumerator:

ChallengeRequest
LoginRequest
LogoutRequest
ExperimentListRequest
ExperimentRequest
PasswordChangeRequest
CreateExperimentRequest
UpdateExperimentRequest
GetListOfOpportunitiesRequest
ScheduleExperimentRequest
ThumbnailRequest
AllThumbnailsRequest
UnscheduleExperimentRequest

Constructor & Destructor Documentation

ServerConnectivity () [private]

~ServerConnectivity () [private]

Member Function Documentation

QString adminUrl () const

Returns:

adminUrl_

void changePassword (QString oldPassword, QString newPassword)
Change password, both oldPassword and newPassword are send to the server. Reply in passwordChanged() (p. 70).
void createExperiment (QString name, int payloadId)
Create a new experiment with name name on payload payloadId. Reply in experimentCreated() (p. 68).

void experimentCreated (int id) [signal]

void experimentListUpdated () [signal]

void experimentReceived (Experiment * experiment) [signal]

void experimentScheduled (bool ok) [signal]

void experimentUnscheduled (bool ok) [signal]

void getAllThumbnails (int experimentId)
Get all thumbnails for experimentId. Replys in multiple newSingleThumbnail() (p. 70) signals.

Todo
No identification of the thumbnail is send with that signal.

void getExperimentDetails (int id)
Retreive experiment details for experiment with id id. Reply in experimentReceived() (p. 68).

QAbstractItemModel* getExperimentListModel () const [inline]
Returns:
Pointer to a model containing the list of experiments.

void getListOfOpportunities (int experimentID, int from, int to)

void getThumbnail (int experimentId)
Get next thumbnail for experimentId. This will not send a reply before a new thumbnail is available. Reply in newThumbnail() (p. 70).

QByteArray hash (QByteArray data) [private]
Returns:
    data hashed using hash(), defined in ../common/utils.h.
CHAPTER 3. COMPONENT DESCRIPTIONS

QByteArray hashedPassword () const

Returns:
hashedPassword_

ServerConnectivity * instance () [static]

Precondition:
i = instance_

Postcondition:
if i = NULL -> instance_. = new ServerConnectivity() (p. 67)

Returns:
instance_

bool interpretBooleanResponse (QByteArray * response, bool * result) [private]

bool interpretChallengeResponse (QByteArray * response) [private]

Todo
Handle errors

bool interpretIntResponse (QByteArray * response, int * result) [private]

bool interpretListOfOpportunities (QByteArray * response, QList< Opportunity > * opportunities) [private]

bool interpretLoginResponse (QByteArray * response, bool * result) [private]

bool interpretOpportunity (QDomElement * e, Opportunity * opportunity) [private]

bool interpretThumbnail (QByteArray * response, Thumbnail * thumbnail) [private]

bool interpretThumbnails (QByteArray * response) [private]

bool isLoggedIn () const [inline]

Returns:
Whether we think we are logged.
CHAPTER 3. COMPONENT DESCRIPTIONS

QString lastErrorMessage () const [inline]

Returns:
Last error message regarding login.

void login ()
When login is executed, a request for a challenge and a session is issued to the server. Once
the requested information is returned, it is used to log in the user with username_ and a hash
of hashedPassword_+challenge When the login is succeeded, a loginFinished() (p. 70) signal is
emitted.

void loginFinished () [signal]

void logout ()

void logoutFinished () [signal]

QHttpRequestHeader makeHttpHeader () [private]

Returns:
 a header suitable for sending XMLRPC requests.

Todo
 Cache this header.

void newSingleThumbnail (Thumbnail * thumbnail) [signal]

void newThumbnail (Thumbnail * thumbnail) [signal]

void opportunitiesReceived (QList< Opportunity > opportunities) [signal]

void passwordChanged (bool ok) [signal]

bool processDetailedExperiment (QByteArray * response, Experiment * experiment) [private]

void processExperiment (QDomElement * e, QStandardItem * payload) [private]

void processExperimentsList (QByteArray * response) [private]

void processPayload (QDomElement * e, QStandardItem * sat) [private]
CHAPTER 3. COMPONENT DESCRIPTIONS

void processPossibleCommand (QDomElement * e, PossibleCommand * cmd) [private]

void processSatellite (QDomElement * e) [private]

void processTimedCommand (QDomElement * e, TimedCommand * cmd) [private]

QString proxyHost () const

Returns:
proxyHost_

quint16 proxyPort () const

Returns:
proxyPost_

bool readBoolean (QDomElement * e, bool * result) [private]
Read a boolean value from XML element e (e->tagName() should be "value") and set result to the value.

Returns:
Whether e contains a valid boolean element.

bool readByteArray (QDomElement * e, QByteArray * result) [private]
Read a base64 encoded blob from XML element e (e->tagName() should be "value") and set result to the decoded value.

Returns:
Whether e contains a valid base64 element.

bool readInt (QDomElement * e, int * result) [private]
Read an integer value from XML element e (e->tagName() should be "value") and set result to the value.

Returns:
Whether e contains a valid string element.
bool readString (QDomElement * e, QString * result) [private]
Read a string value from XML element e (e->tagName() should be "value") and set result to the value.

Returns:
Whether e contains a valid string element.

bool readVariant (QDomElement * e, QVariant * result) [private]
Read a variant value from XML element e (e->tagName() should be "value") and set result to the value. This can be either a boolean, integer or double value.

Returns:
Whether e contains a valid boolean, integer or double element.

void requestFinished (int type, QByteArray * response, bool error) [private, slot]
Todo
Handle failures generically; what to do in case of failure during session?

void resetState ()
Postcondition:
loggedIn = false

void scheduleExperiment (int experimentID, int startTime, QList<Opportunity> opps)

QString serverHost () const
Returns:
serverHost

quint16 serverPort () const
Returns:
serverPort
void setAdminUrl (QString \textit{url})

Parameters:
\begin{itemize}
\item \textit{url} The URL of the administration frontend.
\end{itemize}

Postcondition:
\begin{itemize}
\item adminUrl\_ = url
\end{itemize}

void setPassword (QString \textit{password})

Parameters:
\begin{itemize}
\item \textit{password} The password entered by the user.
\end{itemize}

Postcondition:
\begin{itemize}
\item hashedPassword\_ = sha256(password)
\end{itemize}

void setProxy (QString \textit{host}, quint16 \textit{port})

Parameters:
\begin{itemize}
\item \textit{host} The hostname of the proxy.
\item \textit{port} The port the proxy is listening on.
\end{itemize}

Postcondition:
\begin{itemize}
\item proxyHost\_\ host /\ proxyPort\_ = port /\ The multiHttp is set to host and port
\end{itemize}

Todo
\begin{itemize}
\item Implement user and pass support.
\item Implement SOCKS support.
\end{itemize}

void setServer (QString \textit{host}, quint16 \textit{port})

Parameters:
\begin{itemize}
\item \textit{host} The hostname of the server.
\item \textit{port} The port on which the server listens.
\end{itemize}

Precondition:
\begin{itemize}
\item true
\end{itemize}

Postcondition:
\begin{itemize}
\item serverHost\_ = host /\ serverPort\_ = port /\ multiHttp is set with host and port.
\end{itemize}
void setUsername (QString username)

Parameters:
    username The name of the user.

Postcondition:
    username_ = username

void simpleRequest (int type, QString methodName, QString param = QString())
    [private]
Send a simple XMLRPC request for methodName, with at most one parameter, param. The type (to identify the response) is set to type.

void stop ()

Precondition:
    instance_ != NULL

Postcondition:
    instance_ = NULL

void unscheduleExperiment (int experimentID)

void updateExperiment (Experiment * experiment)
Update experiment with details in experiment. Only fields that are allowed to be changed will be send to the server.

Todo
    Currently no feedback is given.

void updateExperimentList ()

QString username () const

Returns:
    username_

QString usertype () const

Member Data Documentation

QString adminUrl_ [private]
CHAPTER 3. COMPONENT DESCRIPTIONS

QByteArray challenge [private]

QString errorMessage [private]

QStandardItemModel* experimentListModel [private]

QByteArray hashedNewPassword [private]

QByteArray hashedPassword_ [private]

ServerConnectivity * instance_ = 0 [static, private]

Todo
Gracefully handle a dead connection.

bool loggedIn [private]

MultiHttp multiHttp [private]

QString proxyHost_ [private]

quint16 proxyPort_ [private]

QString serverHost_ [private]

quint16 serverPort_ [private]

QByteArray sessionId [private]

QString username_ [private]

QString usertype_ [private]

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

- client/serverconnectivity.h
- client/serverconnectivity.cpp

3.1.34 Thumbnail Class Reference

#include <objects.h>
Public Attributes

- `QByteArray data`
- `int experimentId`
- `int gatheredAt`
- `int id`
- `QString name`
- `QString type`

Detailed Description

Struct representing a thumbnail, including the thumbnail itself.

Member Data Documentation

QByteArray data

int experimentId

int gatheredAt

int id

QString name

QString type

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

• `client/objects.h`

### 3.1.35 TimedCommand Class Reference

#include <objects.h>

Public Attributes

- `int commandId`
- `QVariant paramValue`
- `int time`

Relative time in seconds from start of experiment.

Detailed Description

Struct representing a scheduled command (in a test sequence).
CHAPTER 3. COMPONENT DESCRIPTIONS

Member Data Documentation

int commandId

QVariant paramValue

int time

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

- client/objects.h

3.1.36 UserModel Class Reference

#include <usermodel.h>

Inherits QStandardItemModel.

Inheritance diagram for UserModel:

```
QStandardItemModel

UserModel

UserModel() [inline]
 UserModel() [inline]
```

Public Member Functions

- UserModel (QObject *parent=0)
- ~UserModel ()

Constructor & Destructor Documentation

```
UserModel (QObject * parent = 0) [inline]
~UserModel () [inline]
```

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

- client/usermodel.h

3.1.37 UserSelectionWidget Class Reference

#include <userselectionwidget.h>

Inherits QWidget.
Inheritance diagram for UserSelectionWidget:

![Inheritance diagram]

Public Slots

- void `on_addButton_clicked ()`
- void `on_removeButton_clicked ()`
- void `on_searchUserEdit_textChanged (const QString &text)`

Public Member Functions

- void `setFullsetModel (QStandardItemModel *model)`
- void `setSubsetModel (QStandardItemModel *model)`
- `UserSelectionWidget (QWidget *parent=0)`
- `~UserSelectionWidget ()`

Private Attributes

- QStandardItemModel * `fullsetModel`
- QSortFilterProxyModel * `fullsetProxyModel`
- QStandardItemModel * `subsetModel`
- Ui::UserSelectionWidget `ui`

Constructor & Destructor Documentation

- `UserSelectionWidget (QWidget * parent = 0)`
- `~UserSelectionWidget ()`
Member Function Documentation

void on_addButton_clicked () [slot]

Todo

fix add and remove buttons {

void on_removeButton_clicked () [slot]

void on_searchUserEdit_textChanged (const QString & text) [slot]

void setFullsetModel (QStandardItemModel * model)

void setSubsetModel (QStandardItemModel * model)

Member Data Documentation

QStandardItemModel* fullsetModel [private]

QSortFilterProxyModel* fullsetProxyModel [private]

QStandardItemModel * subsetModel [private]

Ui::UserSelectionWidget ui [private]

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

- client/userselectionwidget.h
- client/userselectionwidget.cpp

3.1.38 client/commands.cpp File Reference

#include <QStandardItemModel>
#include "commands.h"
#include "experimentdialog.h"
3.1.39 client/commands.h File Reference

```cpp
#include <QWidget>
#include <QtDesigner/QDesignerExportWidget>
#include "ui/ui_commands.h"
```

Include dependency graph for commands.h:

This graph shows which files directly or indirectly include this file:

```
client/commands
```

### Classes

- **class Commands**

  A command combined with a time.

3.1.40 client/commandseditdialog.cpp File Reference

```cpp
#include "commandseditdialog.h"
```

Include dependency graph for commandseditdialog.cpp:

```
ui/commandseditdialog
```
3.1.41  client/commandseditdialog.h File Reference

#include <QDialog>
#include "ui/commands_edit.h"

Include dependency graph for commandseditdialog.h:

This graph shows which files directly or indirectly include this file:

Classes

- class CommandsEditDialog
  The dialog in which the commands can be edited.

3.1.42  client/communicationlogsdialog.cpp File Reference

#include "communicationlogsdialog.h"

Include dependency graph for communicationlogsdialog.cpp:

3.1.43  client/communicationlogsdialog.h File Reference

#include <QDialog>
#include "ui_communicationLogs.h"

Include dependency graph for communicationlogsdialog.h:

This graph shows which files directly or indirectly include this file:
CHAPTER 3. COMPONENT DESCRIPTIONS

Classes

- class CommunicationLogsDialog

   The part of the user interface in which communication logs can be viewed.

3.1.44 client/conntest.cpp File Reference

```cpp
#include <iostream>
#include "serverconnectivity.h"
```

Include dependency graph for conntest.cpp:

Functions

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

Function Documentation

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

3.1.45 client/createexperimentdialog.cpp File Reference

```cpp
#include "createexperimentdialog.h"
#include "serverconnectivity.h"
#include <QMessageBox>
```
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for createexperimentdialog.cpp:

3.1.46 client/createexperimentdialog.h File Reference

#include <QDialog>
#include <QModelIndex>
#include ".ui/ui_experiment_create.h"

Include dependency graph for createexperimentdialog.h:

This graph shows which files directly or indirectly include this file:

Classes

- class CreateExperimentDialog
  
The part of the user interface which handles the communication logs.
3.1.47 client/experimentdialog.cpp File Reference

```cpp
#include <QMessageBox>
#include <QCloseEvent>
#include "experimentdialog.h"
#include "manageobservers.h"
#include "serverconnectivity.h"
```

Include dependency graph for experimentdialog.cpp:

3.1.48 client/experimentdialog.h File Reference

```cpp
#include <QMainWindow>
#include <QVariant>
#include "ui_experiment.h"
#include "objects.h"
```

```
#include "ui_manageObservers.h"
```

```cpp
#include "serverconnectivity.h"
#include "multihttp.h"
#include "QObject"
#include "QHttpRequestHeader"
#include "QString"
#include "QByteArray"
#include "QPair"
#include "QVector"
#include "QMap"
```
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for experimentdialog.h:

This graph shows which files directly or indirectly include this file:

Classes

- class ExperimentDialog

3.1.49  client/logindialog.cpp File Reference

```cpp
#include "logindialog.h"
#include "serverconnectivity.h"
#include "preferences.h"
#include <QDebug>
#include <QTimer>
#include <QTimeLine>
#include <QInputDialog>
#include <QMessageBox>
#include <QCloseEvent>
#include <QHttp>
#include <QBuffer>
```
3.1.50 client/logindialog.h File Reference

#include ".ui/ui_login.h"
#include <QDialog>

Include dependency graph for logindialog.h:

This graph shows which files directly or indirectly include this file:

Classes

- class LoginDialog

  Needed for the log in animation.
CHAPTER 3. COMPONENT DESCRIPTIONS

3.1.51 client/main.cpp File Reference

#include "mainwindow.h"
#include <QApplication>
#include <QWidget>

Include dependency graph for main.cpp:

Functions

- int main (int argc, char **argv)

Function Documentation

int main (int argc, char ** argv)

3.1.52 client/mainwindow.cpp File Reference

#include "mainwindow.h"
#include "logindialog.h"
#include "experimentdialog.h"
#include "manageobservers.h"
#include "manageusersdialog.h"
#include "managesatellitesdialog.h"
#include "managepayloadsdialog.h"
#include "observe.h"
#include "preferences.h"
#include "communicationlogsdialog.h"
#include "createexperimentdialog.h"
#include "observationdatadialog.h"
#include "serverconnectivity.h"
#include "selectobservationaldialog.h"
#include "version.h"
#include <QDebug>
#include <QTimer>
#include <QMessageBox>

DDD 0.1 87
#include <QCloseEvent>
#include <QProgressDialog>
#include <QDesktopServices>
#include <QUrl>

Include dependency graph for mainwindow.cpp:

```
ui/ui_mainwindow.h
ui/experiment.h

mainwindow.cpp
mainwindow.h
logindialog.h
eperimentdialog.h
manageobservers.h
manageusersdialog.h
managesatellitesdialog.h
managepayloadsdialog.h
observe.h
objects.h
ui/ui_login.h
ui/ui_manageObservers.h

главна диалог

mainwindow.cpp
mainwindow.h
logindialog.h
experimentdialog.h
manageobservers.h
manageusersdialog.h
managesatellitesdialog.h
managepayloadsdialog.h
observe.h
objects.h
ui/ui_login.h
ui/ui_manageObservers.h
...
```

...
3.1.53  client/mainwindow.h File Reference

#include "ui/mainwindow.h"
#include <QMainWindow>

Include dependency graph for mainwindow.h:

This graph shows which files directly or indirectly include this file:

- class MainWindow
  Needed for the model view controller.

3.1.54  client/manageobservers.cpp File Reference

#include <QDebug>
#include <QStandardItemModel>
#include <QSortFilterProxyModel>
#include "manageobservers.h"

Include dependency graph for manageobservers.cpp:

3.1.55  client/manageobservers.h File Reference

#include <QDialog>
#include "ui/manageObservers.h"
#include "usermodel.h"
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for manageobservers.h:

This graph shows which files directly or indirectly include this file:

Classes

- class ManageObservers

3.1.56 client/managepayloadsdialog.cpp File Reference

#include "managepayloadsdialog.h"
#include ".ui/ui_payloadProperties.h"

Include dependency graph for managepayloadsdialog.cpp:

3.1.57 client/managepayloadsdialog.h File Reference

#include <QDialog>
#include ".ui/ui_managePayloads.h"

Include dependency graph for managepayloadsdialog.h:
CHAPTER 3. COMPONENT DESCRIPTIONS

This graph shows which files directly or indirectly include this file:

```
client/managepayloadsdialog.h
client/mainwindow.cpp
client/managepayloadsdialog.cpp
```

Classes

- class ManagePayloadsDialog

3.1.58 client/managesatellitesdialog.cpp File Reference

```cpp
#include "managesatellitesdialog.h"
#include ".ui/ui_satelliteProperties.h"
```

Include dependency graph for managesatellitesdialog.cpp:

```
QDialog
ui/manageSatellites
```

3.1.59 client/managesatellitesdialog.h File Reference

```cpp
#include <QDialog>
#include "ui_manageSatellites.h"
```

Include dependency graph for managesatellitesdialog.h:

```
QDialog
ui/manageSatellites
```

This graph shows which files directly or indirectly include this file:

```
client/mainwindow.cpp
client/managesatellitesdialog.cpp
```

Classes

- class ManageSatellitesDialog

3.1.60 client/manageusersdialog.cpp File Reference

```cpp
#include "manageusersdialog.h"
```
#include "ui/userProperties.h"

Include dependency graph for manageusersdialog.cpp:

3.1.61 client/manageusersdialog.h File Reference

#include <QDialog>
#include "ui/manageUsers.h"
#include "ui_userProperties.h"

Include dependency graph for manageusersdialog.h:

This graph shows which files directly or indirectly include this file:

Classes

- class ManageUsersDialog

3.1.62 client/multihttp.cpp File Reference

#include "multihttp.h"
#include <QByteArray>
#include <QHttp>
#include <QBuffer>
#include <QDebug>
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for multihttp.cpp:

```
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for multihttp.cpp:

3.1.63 client/multihttp.h File Reference

#include <QObject>
#include <QVector>
#include <QMap>
#include <QHttpRequestHeader>

Include dependency graph for multihttp.h:

3.1.63 client/multihttp.h File Reference

#include <QObject>
#include <QVector>
#include <QMap>
#include <QHttpRequestHeader>

Include dependency graph for multihttp.h:
CHAPTER 3. COMPONENT DESCRIPTIONS

This graph shows which files directly or indirectly include this file:

![Dependency Graph]

**Classes**

- **class MultiHttp**

### 3.1.64 client/objects.h File Reference

```cpp
#include <QVariant>
```

Include dependency graph for objects.h:

![Dependency Graph for objects.h]
CHAPTER 3. COMPONENT DESCRIPTIONS

This graph shows which files directly or indirectly include this file:

```
Classes

- class Experiment
- class Opportunity
- class PossibleCommand
- class Thumbnail
- class TimedCommand
```

3.1.65 client/observationdatadialog.cpp File Reference

```cpp
#include "observationdatadialog.h"
#include ".ui/ui_highResolutionData.h"
#include "serverconnectivity.h"
#include "objects.h"
#include <QStandardItemModel>
#include <QAction>
```
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for observationdatadialog.cpp:

3.1.66 client/observationdatadialog.h File Reference

```
#include <QDialog>
#include <QModelIndex>
#include "ui_observationData.h"
```

Include dependency graph for observationdatadialog.h:

This graph shows which files directly or indirectly include this file:

Classes

- class ObservationDataDialog

3.1.67 client/observationview.cpp File Reference

```
#include "observationview.h"
```
CHAPTER 3. COMPONENT DESCRIPTIONS

3.1.68 client/observationview.h File Reference

#include ".ui/ui_observationview.h"
#include <QWidget>
#include <QtDesigner/QDesignerExportWidget>

Include dependency graph for observationview.h:

This graph shows which files directly or indirectly include this file:

Classes

- class ObservationView

3.1.69 client/observe.cpp File Reference

#include "observe.h"
#include "serverconnectivity.h"
#include <QCloseEvent>
#include <QPixmap>
#include <QDateTime>
#include <QDebug>
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for observe.cpp:

```
3.1.70 client/observe.h File Reference

#include <QMainWindow>
#include <QModelIndex>
#include "ui_observe.h"
#include "objects.h"

This graph shows which files directly or indirectly include this file:

Classes

- class Observe
  Provides functionality for observing an experiment to the client.
3.1.71 client/preferences.cpp File Reference

```cpp
#include "preferences.h"
#include "serverconnectivity.h"
#include <QMessageBox>
#include <QDebug>
```

Include dependency graph for preferences.cpp:

3.1.72 client/preferences.h File Reference

```cpp
#include <QDialog>
#include "ui_preferences.h"
```

Include dependency graph for preferences.h:

This graph shows which files directly or indirectly include this file:
CHAPTER 3. COMPONENT DESCRIPTIONS

Classes

- class Preferences

3.1.73 client/selectobservedialog.cpp File Reference

#include "selectobservedialog.h"
#include "serverconnectivity.h"
#include <QDebug>
#include <QPushButton>

Include dependency graph for selectobservedialog.cpp:

3.1.74 client/selectobservedialog.h File Reference

#include <QDialog>
#include <QModelIndex>
#include "ui_selectObserve.h"

Include dependency graph for selectobservedialog.h:
CHAPTER 3. COMPONENT DESCRIPTIONS

This graph shows which files directly or indirectly include this file:

```
client/selectobservedialog.h
client/mainwindow.cpp
client/selectobservedialog.cpp
```

3.1.75 client/serverconnectivity.cpp File Reference

```cpp
#include "serverconnectivity.h"
#include "version.h"
#include "./common/utils.h"
#include <QDebug>
#include <QHttp>
#include <QBuffer>
#include <QTimer>
#include <QSettings>
#include <QCoreApplication>
#include <QDomDocument>
#include <QStandardItemModel>
```
Include dependency graph for serverconnectivity.cpp:

3.1.76 client/serverconnectivity.h File Reference

```
#include "multihttp.h"
#include "objects.h"
#include <QString>
#include <QObject>
#include <QByteArray>
#include <QPair>
#include <QStandardItemModel>
#include <QHttpRequestHeader>
```
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for serverconnectivity.h:

This graph shows which files directly or indirectly include this file:

Classes

- class ServerConnectivity

3.1.77 client/usermodel.h File Reference

#include <QStandardItemModel>

Include dependency graph for usermodel.h:
CHAPTER 3. COMPONENT DESCRIPTIONS

This graph shows which files directly or indirectly include this file:

- client/usermodel.h
- client/manageobservers.h
- client/experimentdialog.cpp
- client/mainwindow.cpp
- client/manageobservers.cpp

Classes

- class UserModel

### 3.1.78 client/userselectionwidget.cpp File Reference

```c
#include "userselectionwidget.h"
#include <QStandardItemModel>
#include <QSortFilterProxyModel>
#include <QDebug>
```

Include dependency graph for userselectionwidget.cpp:

### 3.1.79 client/userselectionwidget.h File Reference

```c
#include <QWidget>
#include "ui_userSelection.h"
```

Include dependency graph for userselectionwidget.h:

This graph shows which files directly or indirectly include this file:

- client/userselectionwidget.h
- client/userselectionwidget.cpp
CHAPTER 3. COMPONENT DESCRIPTIONS

Classes

- class UserSelectionWidget

3.1.80 client/version.h File Reference

This graph shows which files directly or indirectly include this file:

![Graph showing file dependencies]

Defines

- #define IMSETY_VERSION "0.2.1"

Define Documentation

#define IMSETY_VERSION "0.2.1"

3.1.81 Todo List

Member Commands::Commands (p. 12) Values are not restricted to be in the min-max range.

Member Commands::setupCommands (p. 14) Sort commands

Member ExperimentDialog::ExperimentDialog (p. 25) Invent some userfriendly way of blocking access until data is loaded.

Member ExperimentDialog::addTimedCommand (p. 25) Setting this to true will work but causes some unwanted effects. Implement moving of commands in the list.

Member ExperimentDialog::addTimedCommand (p. 25) What will happen if isValid ?

Member ExperimentDialog::on_commandAddButton_clicked (p. 27) Disable add button when no command selected.

Member ExperimentDialog::on_observersButton_clicked (p. 28) Implement ManageObservers (p. 38) dialog.

Member LoginDialog::on_loginButton_clicked (p. 32) Make a save password checkbox.
Member MainWindow::closeEvent (p. 36)  Show messagebox if QApplication::topLevelWidgets .count > 1?

Member MainWindow::selectionChanged (p. 38)  Not yet implemented: delete experiment function.

Member MainWindow::selectionChanged (p. 38)  Not yet implemented: manage observers for an experiment.

Member ManageObservers::ManageObservers (p. 39)  Check for duplicates when copying users.

Member Observe::Observe (p. 51)  Set name of groundstation or other info.

Member Observe::Observe (p. 51)  Set name of groundstation or other info.

Member Observe::newThumbnail (p. 52)  Make waiting time settable via slider in UI? This would allow a scientist to slow down the image flow to look more carefully at the experiment results.

Member Preferences::minPasswordLength (p. 55)  Fix behaviour when enter is pressed in password boxes.

Member ServerConnectivity::instance_ (p. 75)  Gracefully handle a dead connection.

Member ServerConnectivity::getAllThumbnails (p. 68)  No identification of the thumbnail is send with that signal.

Member ServerConnectivity::interpretChallengeResponse (p. 69)  Handle errors

Member ServerConnectivity::makeHttpHeader (p. 70)  Cache this header.

Member ServerConnectivity::requestFinished (p. 72)  Handle failures generically; what to do in case of failure during session?

Member ServerConnectivity::setProxy (p. 73)  Implement user and pass support. Implement SOCKS support.

Member ServerConnectivity::updateExperiment (p. 74)  Currently no feedback is given.

Member UserSelectionWidget::on_addButton_clicked (p. 79)  fix add and remove buttons
CHAPTER 3. COMPONENT DESCRIPTIONS

3.2 Server

3.2.1 Future work

Most of the server has been finished conforming to the design as described in the ADD [3]. The major part that is currently not implemented is the functionality needed to facilitate the administration to the client. This functionality has been implemented at the administration level, but still has to be implemented in the Controller and Client connectivity components. Stubs for this functionality have been added in the Controller with their interface description. The issues can be found in the todo list of the server (section 3.2.156).

Furthermore there are a couple of minor issues. First of all the handling of timeouts in the scheduler (being timeouts in communication with MCCs and MCSes needs to be handled in a better way). Secondly the timing out of these should be reported to the controller such that threads waiting for a reaction from the scheduler here can be terminated.

3.2.2 IMSETY server Directories

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

common ................................................................. 112
server ................................................................. 113

3.2.3 IMSETY server Namespace List

Here is a list of all namespaces with brief descriptions:

std ................................................................. 114

3.2.4 IMSETY server Class Hierarchy

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

Account ................................................................. 114
AdminLowlevel .......................................................... 126
DatabaseAbstraction .................................................... 210
Booking ................................................................. 150
BookingInfo ............................................................. 157
Command ............................................................... 162
CommandParam .......................................................... 168
ConnectionManager ..................................................... 169
Controller ............................................................... 170
Controller::CallbackHandleOpportunities ......................... 193
Controller::ExperimentInfo .......................................... 195
Controller::PayloadInfo ............................................... 195
Controller::SessionInfo .............................................. 196
Controller::ThreadArgumentInt ..................................... 197
Controller::ThreadArgumentStringInt ......................... 197
Controller::ThreadingInformation .................................. 197
Data ................................................................. 199
Experiment ............................................................. 221
ExperimentCommand .................................................. 232

DDD 0.1 107
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ExperimentPart</strong></td>
<td>233</td>
</tr>
<tr>
<td><strong>Interval</strong></td>
<td>244</td>
</tr>
<tr>
<td><strong>MccInfo</strong></td>
<td>257</td>
</tr>
<tr>
<td><strong>McsInfo</strong></td>
<td>272</td>
</tr>
<tr>
<td><strong>ParamTypes</strong></td>
<td>275</td>
</tr>
<tr>
<td><strong>Payload</strong></td>
<td>276</td>
</tr>
<tr>
<td><strong>Queue</strong></td>
<td>283</td>
</tr>
<tr>
<td><strong>QueueItem</strong></td>
<td>289</td>
</tr>
<tr>
<td><strong>Queues</strong></td>
<td>290</td>
</tr>
<tr>
<td><strong>Satellite</strong></td>
<td>293</td>
</tr>
<tr>
<td><strong>SchedInfo</strong></td>
<td>300</td>
</tr>
<tr>
<td><strong>Scheduler</strong></td>
<td>302</td>
</tr>
<tr>
<td><strong>sha2</strong></td>
<td>314</td>
</tr>
<tr>
<td><strong>sha2::SHA_CTX</strong></td>
<td>318</td>
</tr>
<tr>
<td><strong>SingleLog</strong></td>
<td>318</td>
</tr>
<tr>
<td><strong>SocketInfo</strong></td>
<td>320</td>
</tr>
<tr>
<td><strong>runtime_error</strong></td>
<td>320</td>
</tr>
<tr>
<td><strong>Account::AccountException</strong></td>
<td>125</td>
</tr>
<tr>
<td><strong>AdminLowlevel::AdminLowlevelException</strong></td>
<td>148</td>
</tr>
<tr>
<td><strong>DatabaseAbstraction::DatabaseException</strong></td>
<td>220</td>
</tr>
<tr>
<td><strong>Booking::BookingException</strong></td>
<td>157</td>
</tr>
<tr>
<td><strong>Command::CommandException</strong></td>
<td>168</td>
</tr>
<tr>
<td><strong>Controller::ControllerException</strong></td>
<td>194</td>
</tr>
<tr>
<td><strong>Data::DataException</strong></td>
<td>209</td>
</tr>
<tr>
<td><strong>Experiment::ExperimentException</strong></td>
<td>231</td>
</tr>
<tr>
<td><strong>LogException</strong></td>
<td>247</td>
</tr>
<tr>
<td><strong>Payload::PayloadException</strong></td>
<td>282</td>
</tr>
<tr>
<td><strong>Queue::QueueException</strong></td>
<td>288</td>
</tr>
<tr>
<td><strong>Satellite::SatelliteException</strong></td>
<td>299</td>
</tr>
<tr>
<td><strong>Scheduler::SchedulerException</strong></td>
<td>313</td>
</tr>
<tr>
<td><strong>XmlRpcController::XmlRpcControllerException</strong></td>
<td>328</td>
</tr>
<tr>
<td><strong>xmlrpc::method</strong></td>
<td>323</td>
</tr>
<tr>
<td><strong>acknowledged</strong></td>
<td>125</td>
</tr>
<tr>
<td><strong>booked</strong></td>
<td>149</td>
</tr>
<tr>
<td><strong>cancelled</strong></td>
<td>158</td>
</tr>
<tr>
<td><strong>changePassword</strong></td>
<td>159</td>
</tr>
<tr>
<td><strong>createExperiment</strong></td>
<td>198</td>
</tr>
<tr>
<td><strong>failed</strong></td>
<td>233</td>
</tr>
<tr>
<td><strong>getChallengeAndSession</strong></td>
<td>234</td>
</tr>
<tr>
<td><strong>getExperiment</strong></td>
<td>236</td>
</tr>
<tr>
<td><strong>getExperiments</strong></td>
<td>237</td>
</tr>
<tr>
<td><strong>getListOfOpportunities</strong></td>
<td>238</td>
</tr>
<tr>
<td><strong>getListOfThumbnails</strong></td>
<td>239</td>
</tr>
<tr>
<td><strong>getPayload</strong></td>
<td>240</td>
</tr>
<tr>
<td><strong>getSatellite</strong></td>
<td>242</td>
</tr>
<tr>
<td><strong>getThumbnail</strong></td>
<td>243</td>
</tr>
<tr>
<td><strong>listFailure</strong></td>
<td>245</td>
</tr>
<tr>
<td><strong>listOfOpportunities</strong></td>
<td>246</td>
</tr>
<tr>
<td><strong>login</strong></td>
<td>247</td>
</tr>
<tr>
<td><strong>logout</strong></td>
<td>249</td>
</tr>
<tr>
<td><strong>newData</strong></td>
<td>274</td>
</tr>
<tr>
<td><strong>requestFailure</strong></td>
<td>291</td>
</tr>
<tr>
<td><strong>requestRejected</strong></td>
<td>292</td>
</tr>
</tbody>
</table>
CHAPTER 3. COMPONENT DESCRIPTIONS

3.2.5 IMSETY server Class List

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

- **Account (< std::exception>)**
- **Account::AccountException**
- **acknowledged**
- **AdminLowlevel** (Class that presents an administration interface to the lower layers of the system)
- **AdminLowlevel::AdminLowlevelException** (Exception class thrown by AdminLowlevel (p. 126))
- **booked**
- **Booking**
- **Booking::BookingException**
- **BookingInfo** (Specifies a booking)
- **cancelled**
- **changePassword** (Changes the password of an account)
- **ClientConnectivity**
- **Command (< std::exception>)**
- **Command::CommandException** (Exception class thrown by Command (p. 162))
- **CommandParam** (Parameters of a command which can be used in an experiment)
- **ConnectionManager** (Provides a connection to the lower layer in order to abstract from the actual connection)
- **Controller::CallbackHandleOpportunities** (Callback handle for requesting list of opportunities)
- **Controller::ControllerException** (Exception thrown by the Controller (p. 170) class)
- **Controller::ExperimentInfo** (Class used in the return value of getExperimentIDs)
- **Controller::PayloadInfo** (Class used in the return value of getPayloadIDs)
- **Controller::SessionInfo** (Simple structure that is used to keep the information belonging to a session)
- **Controller::ThreadArgumentInt** (Structure used to pass controller and id into a thread)
- **Controller::ThreadArgumentStringInt** (Structure used to pass controller, id and session into a thread)
- **Controller::ThreadingInformation** (Administration for threads)
- **createExperiment** (Creates a new experiment for a payload with payloadID and with a name)
Data (< std::exception Administration frontend for managing data entities ) 

Data::DataBaseException (Exception class thrown by Data (p. 199 ))

DatabaseAbstraction (< Type definition for SatelliteID )

DatabaseAbstraction::DatabaseException

Experiment (< std::utility Administration frontend for managing experiment entities )

Experiment::ExperimentException

ExperimentCommand (Parameters of a command as used in an experiment )

ExperimentPart (Part of an experiment, as used in on MCC/MCS and scheduler )

failed

catch (Exception thrown by the scheduler )

Scheduler::SchedulerException

Scheduler (A class that contains information needed by the thread that checks a satellite )

SchedInfo (Exception class thrown by Satellite::SatelliteException)

Satellite (Request to schedule an experiment )

Satellite::SatelliteException

Satellite (Specify a queue item, as used in queues )

Queue::QueueException

Queue (Exception class thrown by Payload::PayloadException)

Payload (List of possible types for the parameters of a command )

Payload::PayloadException

Login (Login remote procedure call )

Logout (Logout remote procedure call )

MccController (This class manages all connections with the MCCs )

MccController::MccControllerException

MccController (This class contains information for communication with a MCC )

MccController::MccControllerException

MccController (This class manages all connections with the MCCs )

MccController::MccControllerException

MccController (This class contains information for communication with an MCS )

MccController::MccControllerException

MccController (This class can be used to send messages to a MCC )

MccController::MccControllerException

MccController (This class can be used to send messages to a MCS )

MccController::MccControllerException

McsClient (This class contains information for communication with an MCS )

McsClient::McsControllerException

McsClient (This class contains information for communication with a MCC )

McsClient::McsControllerException

McsController (This class can be used to send messages to a MCS )

McsController::McsControllerException

McsController (This class contains information for communication with a MCC )

McsController::McsControllerException

MccController (This class can be used to receive messages from a MCC )

MccController::MccControllerException

MccController (This class can be used to receive messages from a MCC )

MccController::MccControllerException

getChallengeAndSession (Returns a challenge and a session, to be used by the user to identify himself to the server )

catch (Get experiment by identifier )

getExperiments (Gets all satellites, payloads and experiments associated with an account )

catch (Get experiment by identifier )

getListOfOpportunities (Returns a list of opportunities for an experiment with experimentID )

catch (Get experiment by identifier )

getListOfThumbnails (Requests a list of thumbnails )

catch (Get experiment by identifier )

getPayload (Get payload by identifier )

catch (Get experiment by identifier )

getSatellite (Get satellite by identifier )

catch (Get experiment by identifier )

getThumbnail (Get the next observation thumbnail for an experiment )

Interval (Specifies an interval in time, to be used in a booking )

catch (Get experiment by identifier )

listFailure

listOfOpportunities

catch (Get experiment by identifier )

LogException (< std::exception )

login (Login remote procedure call )

logout (Logout remote procedure call )

MccClient (This class can be used to send messages to a MCC )

MccClient::MccControllerException

McsClient (This class can be used to send messages to a MCS )

McsClient::McsControllerException

McsController (This class manages all connections with the MCSs )

McsController::McsControllerException

McsController (This class contains information for communication with the MCCs )

McsController::McsControllerException

newData

catch (Get experiment by identifier )

ParamTypes (List of possible types for the parameters of a command )

catch (Get experiment by identifier )

Payload (Exception class thrown by Payload (p. 276 ))

catch (Get experiment by identifier )

Queue (Exception class thrown by Satellite::SatelliteException)

catch (Get experiment by identifier )

Queue::QueueException

QueueItem (Specifies a queue item, as used in queues )

catch (Get experiment by identifier )

Queues (Class that contains an internal and an external queue )

catch (Get experiment by identifier )

requestFailure

catch (Get experiment by identifier )

requestRejected

catch (Get experiment by identifier )

Satellite (Exception class thrown by Satellite (p. 293 ))

catch (Get experiment by identifier )

Satellite::SatelliteException

Satellite (List of possible types for the parameters of a command )

catch (Get experiment by identifier )

Satellite::SatelliteException

SchedInfo (A class that contains information needed by the thread that checks a satellite)

scheduleExperiment (Requests to schedule an experiment )

Schedule

Scheduler::SchedulerException (Exception thrown by the scheduler )

sha2

sha2::SHA_CTX

SingleLog (Specifies a log item )
### CHAPTER 3. COMPONENT DESCRIPTIONS

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SocketInfo</td>
<td>Convenience struct</td>
<td>320</td>
</tr>
<tr>
<td>runtime_error</td>
<td></td>
<td>320</td>
</tr>
<tr>
<td>unscheduleExperiment</td>
<td>Requests to unschedule an experiment</td>
<td>321</td>
</tr>
<tr>
<td>updateExperiment</td>
<td>Updates the experiment on the server</td>
<td>322</td>
</tr>
<tr>
<td>xmlrpc_c::method</td>
<td></td>
<td>323</td>
</tr>
<tr>
<td>XmlRpcClient</td>
<td></td>
<td>324</td>
</tr>
<tr>
<td>XmlRpcController</td>
<td>This class manages all connections with XML RPC servers</td>
<td>325</td>
</tr>
<tr>
<td>XmlRpcController::XmlRpcControllerException</td>
<td>Exception class thrown by the Controller (p. 170)</td>
<td>328</td>
</tr>
<tr>
<td>XmlRpcListener</td>
<td>This class can be used to receive messages from a XML RPC server</td>
<td>328</td>
</tr>
</tbody>
</table>

#### 3.2.6 IMSETTY server File List

Here is a list of all files with brief descriptions:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>common/account.cpp</td>
<td>330</td>
</tr>
<tr>
<td>common/account.h</td>
<td>330</td>
</tr>
<tr>
<td>common/adminlowlevel.h</td>
<td>331</td>
</tr>
<tr>
<td>common/command.cpp</td>
<td>332</td>
</tr>
<tr>
<td>common/command.h</td>
<td>333</td>
</tr>
<tr>
<td>common/connectionmanager.cpp</td>
<td>334</td>
</tr>
<tr>
<td>common/connectionmanager.h</td>
<td>334</td>
</tr>
<tr>
<td>common/data.cpp</td>
<td>335</td>
</tr>
<tr>
<td>common/data.h</td>
<td>336</td>
</tr>
<tr>
<td>common/defs.h</td>
<td>338</td>
</tr>
<tr>
<td>common/experiment.cpp</td>
<td>344</td>
</tr>
<tr>
<td>common/experiment.h</td>
<td>345</td>
</tr>
<tr>
<td>common/log.cpp</td>
<td>347</td>
</tr>
<tr>
<td>common/log.h</td>
<td>350</td>
</tr>
<tr>
<td>common/payload.cpp</td>
<td>354</td>
</tr>
<tr>
<td>common/payload.h</td>
<td>355</td>
</tr>
<tr>
<td>common/satellite.cpp</td>
<td>357</td>
</tr>
<tr>
<td>common/satellite.h</td>
<td>358</td>
</tr>
<tr>
<td>common/sha2.cpp</td>
<td>359</td>
</tr>
<tr>
<td>common/sha2.h</td>
<td>361</td>
</tr>
<tr>
<td>common/utils.h</td>
<td>363</td>
</tr>
<tr>
<td>server/booking.cpp</td>
<td>364</td>
</tr>
<tr>
<td>server/booking.h</td>
<td>366</td>
</tr>
<tr>
<td>server/clientconnectivity.cpp</td>
<td>369</td>
</tr>
<tr>
<td>server/clientconnectivity.h</td>
<td>371</td>
</tr>
<tr>
<td>server/controller.cpp</td>
<td>372</td>
</tr>
<tr>
<td>server/controller.h</td>
<td>374</td>
</tr>
<tr>
<td>server/databaseabstraction.cpp</td>
<td>377</td>
</tr>
<tr>
<td>server/databaseabstraction.h</td>
<td>377</td>
</tr>
<tr>
<td>server/main.cpp</td>
<td>378</td>
</tr>
<tr>
<td>server/mccClient.cpp</td>
<td>380</td>
</tr>
<tr>
<td>server/mccClient.h</td>
<td>380</td>
</tr>
<tr>
<td>server/mccController.cpp</td>
<td>381</td>
</tr>
<tr>
<td>server/mccController.h</td>
<td>382</td>
</tr>
<tr>
<td>server/mccGeneral.cpp</td>
<td>384</td>
</tr>
<tr>
<td>server/mccGeneral.h</td>
<td>385</td>
</tr>
<tr>
<td>server/mccListener.cpp</td>
<td>387</td>
</tr>
<tr>
<td>server/mccListener.h</td>
<td>388</td>
</tr>
</tbody>
</table>
3.2.7 IMSETY server Related Pages

Here is a list of all related documentation pages:

Todo List ................................................. 417

3.2.8 common/ Directory Reference

Files

- file account.cpp
- file account.h
- file adminlowlevel.h
- file command.cpp
- file command.h
- file connectionmanager.cpp
- file connectionmanager.h
- file data.cpp
CHAPTER 3. COMPONENT DESCRIPTIONS

- file data.h
- file defs.h
- file experiment.cpp
- file experiment.h
- file log.cpp
- file log.h
- file payload.cpp
- file payload.h
- file satellite.cpp
- file satellite.h
- file sha2.cpp
- file sha2.h
- file utils.h

3.2.9 server/ Directory Reference

Files

- file booking.cpp
- file booking.h
- file clientconnectivity.cpp
- file clientconnectivity.h
- file controller.cpp
- file controller.h
- file databaseabstraction.cpp
- file databaseabstraction.h
- file main.cpp
- file mccClient.cpp
- file mccClient.h
- file mccController.cpp
- file mccController.h
- file mccGeneral.cpp
- file mccGeneral.h
- file mccListener.cpp
- file mccListener.h
- file mcsClient.cpp
- file mcsClient.h
- file mcsController.cpp
- file mcsController.h
- file mcsGeneral.cpp
3.2.10 std Namespace Reference

3.2.11 Account Class Reference

```cpp
#include <account.h>
```

Public Member Functions

- **Account ()**
  
  *Empty constructor.*

- **Account (const AccountID id) throw (AccountException)**

  *Constructor of a new account object with accountid username.*

- **Account (const AccountID id, const AccountType &anAccountType, const std::string &aPassword, const std::string &fullName, const std::string &email) throw (AccountException)**

  *Constructor of a new account object, which is not yet in the database.*

- **bool accountExists () throw (AccountException)**

  *Checks whether the current account exists in the database.*

- **void addExperiment (const ExperimentID id) throw (AccountException)**

  *Gives the account access to an experiment.*

- **void addPayload (const PayloadID id) throw (AccountException)**

  *Give this account access to a payload.*

- **std::string getEmail () throw (AccountException)**

  *Retrieves the email adress.*
std::string getFullName () throw (AccountException)
Retrieves the full name.

AccountID getID () const
Returns the identifier for this account.

std::string getPassword () throw (AccountException)
Retrieves the password.

AccountType getType () throw (AccountException)
Return the type of the account.

bool hasRightsToExperiment (const ExperimentID id) throw (AccountException)
Returns whether this account has access to the experiment with eID.

bool hasRightsToPayload (const PayloadID id) throw (AccountException)
Returns whether this account has access to the payload with pID.

bool hasRightsToSatellite (const SatelliteID id) throw (AccountException)
Returns whether this account has access to the satellite with sID.

void instantiate () throw (AccountException)
Retrieves data from account with accountID from lower layer.

bool isRegistered () throw (AccountException)
Returns where the account is registered or not.

void registerAccount ()
Registers the account.

void removeExperiment (const ExperimentID id) throw (AccountException)
Removes the rights to access an experiment.

void removePayload (const PayloadID id) throw (AccountException)
Remove the rights to access to a payload.

void setEmail (const std::string &email)
Sets an email for the account.

void setFieldName (const std::string &fullName)
Sets a fullname for the account.

void setPassword (const std::string &aPassword)
Sets a password for the account.

void setType (const AccountType anAccountType)
Set the type of the account.

void setUpdate ()
Indicate that the data needs to be updated.

- void `store` () throw (AccountException)
  Stores the data of the account into the database.

- void `unregisterAccount` ()
  Unregisters the account.

- ~Account ()
  Destructor of account object.

### Private Attributes

- std::string `email`
  the emailaddress of the user

- std::string `fullname`
  the fullname of the user

- AccountID `identifier`
  Identifier for this object.

- bool `needs_storage`
- bool `needs_update`
- std::string `password`
  the password to access the account.

- bool `registered`
  Whether this account is registered.

- AccountType `type`
  The account type.

### Classes

- class `AccountException`

### Detailed Description

**Account**

(const AccountID `id`, const AccountType & `anAccountType`, const std::string & `aPassword`, const std::string & `fullName`, const std::string & `email`) throw (AccountException)
CHAPTER 3. COMPONENT DESCRIPTIONS

Exceptions:

AccountException (p. 125)

Parameters:

id the account identifier.

anAccountType The account type.

aPassword The password for a nonadmin user to access the account.

fullName The full name of the user for this account.

eMail The e-mail address of the user for this account.

Precondition:

anAccountId <> "" \ (exists no AccountID’ in Account (p. 114) (AccountID’ is the same as anAccountID)).

Postcondition:

this.getID() = anAccountId, this.getType() = anAccountType, this.getPassword() = aPassword,
this.registered = false, this.needs_update = false, this.needs_storage = false, this.fullname
= fullName, this.email = email.

Account (const AccountID id) throw (AccountException)

Parameters:

id an AccountID identifying the account in the lower layer.

Precondition:

true

Postcondition:

this.getID() = id.

Account ()

Precondition:

true

Postcondition:

this.identifier = "" this.type = Observer this.password= "" this.registered = false this.needs_update = false this.needs_storage = false this.fullname = "" this.email = ""

~Account ()
Member Function Documentation

bool accountExists () throw (AccountException)

Exceptions:

AccountException (p. 125)

Precondition:

ture

Returns:

(exists a in Account (p. 114) ( a.identifier = this.getID)).

void addExperiment (const ExperimentID id) throw (AccountException)

Exceptions:

AccountException (p. 125)

Parameters:

id an experiment identifier which will be given access to.

Precondition:

this.registered = true

Postcondition:

this.hasRightsToExperiment(id) = true

void addPayload (const PayloadID id) throw (AccountException)

Exceptions:

AccountException (p. 125)

Parameters:

id A payload Identifier which will be given access to.

Precondition:

this.registered = true

Postcondition:

this.hasRightsToPayload(id) \ (Forall e in Experiment (p. 221) : (e.getPayload() = id =>
this.hasRightsToExperiment(e))))
std::string getEmail () throw (AccountException)

Exceptions:

AccountException (p. 125)

Precondition:

true

true

Returns:

this.email

std::string getFullName () throw (AccountException)

Exceptions:

AccountException (p. 125)

Precondition:

true

true

Returns:

this.fullname

AccountID getID () const

Precondition:

true

Returns:

identifier

std::string getPassword () throw (AccountException)

Exceptions:

AccountException (p. 125)

Precondition:

true

Returns:

this.password
CHAPTER 3. COMPONENT DESCRIPTIONS

AccountType getType () throw (AccountException)

Exceptions:

AccountException (p. 125)

Precondition:

true

Returns:

this.type

bool hasRightsToExperiment (const ExperimentID id) throw (AccountException)

Exceptions:

AccountException (p. 125)

Parameters:

id An experiment Identifier.

Precondition:

true

Returns:

Exists an id’ in Experiment (p. 221): id’ = id and this.canObserve = id It should be noted that an Administrator has rights to all experiments.

bool hasRightsToPayload (const PayloadID id) throw (AccountException)

Exceptions:

AccountException (p. 125)

Parameters:

id A payload identifier.

Precondition:

true

Returns:

Exists a id’ in Payload (p. 276): id’ = id and this.hasAccessTo = id. It should be noted that an Administrator has rights to all payloads.
bool hasRightsToSatellite (const SatelliteID id) throw (AccountException)

Exceptions:

(AccountException (p. 125)

Parameters:

id A satellite identifier.

Precondition:

true

Returns:

Whether this user has access to a Payload (p. 276) or an Experiment (p. 221) defined on this satellite. It should be noted that an Administrator has rights to all satellites.

void instantiate () throw (AccountException)

Exceptions:

(AccountException (p. 125)

Precondition:

identifier <> ""

Postcondition:

The parameters of account are instantiated with information from the lower layer.

bool isRegistered () throw (AccountException)

Exceptions:

(AccountException (p. 125)

Precondition:

true

Returns:

this.registered

void registerAccount ()

Precondition:

true

Postcondition:

this.registered = true
void removeExperiment (const ExperimentID id) throw (AccountException)

Exceptions:

AccountException (p. 125)

Parameters:

id an experiment identifier which will be removed access from.

Precondition:

this.registered = true

Postcondition:

this.hasRightsToExperiment(id) = false

void removePayload (const PayloadID id) throw (AccountException)

Exceptions:

AccountException (p. 125)

Parameters:

id a payload identifier which will be removed access from.

Precondition:

this.registered = true

Postcondition:

this.hasRightToPayload(id) = false /
(Forall e in Experiment (p. 221) : (e.getPayload() = id => not(this.hasRightsToExperiment(e))))

void setEmail (const std::string & email)

Parameters:

e-mail A new emailadress for the account.

Precondition:

true

Postcondition:

this.email = email
void setFullName (const std::string & fullName)

Parameters:
    fullName A new full name for the account.

Precondition:
    true

Postcondition:
    this.fullname = fullName

void setPassword (const std::string & aPassword)

Parameters:
    aPassword A new password for the account.

Precondition:
    true

Postcondition:
    this.password = aPassword

void setType (const AccountType anAccountType)

Parameters:
    anAccountType The new account type.

Precondition:
    true

Postcondition:
    this.type = anAccountType

void setUpdate ()

Precondition:
    this.identifier > -1

Postcondition:
    this.needs_update = true
void store () throw (AccountException)

Exceptions:

AccountException (p. 125)

Precondition:
true

Postcondition:
The data of the account is stored into the database if the data is changed.

void unregisterAccount ()

Precondition:
true

Postcondition:
this.registered = false

Member Data Documentation

std::string email [private]

std::string fullname [private]

AccountID identifier [private]

bool needs_storage [private]
Whether the data of this payload needs client-server communication to store its attributes

bool needs_update [private]
Whether the data of this payload needs client-server communication to update its attributes

std::string password [private]

bool registered [private]

AccountType type [private]

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

- common/account.h
- common/account.cpp
3.2.12 Account::AccountException Class Reference

#include <account.h>

Inherits `runtime_error`.

Inheritance diagram for Account::AccountException:

![Inheritance diagram for Account::AccountException](image)

Public Member Functions

- **AccountException** (const std::string &arg)

Constructor & Destructor Documentation

**AccountException** (const std::string & arg) [inline]

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

- common/account.h

3.2.13 acknowledged Class Reference

Inherits `xmlrpc_c::method`.

Inheritance diagram for acknowledged:

![Inheritance diagram for acknowledged](image)

Public Member Functions

- **acknowledged** (Satellite *tmpSat, McsController *tmpController)
- **void execute** (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
CHAPTER 3. COMPONENT DESCRIPTIONS

Private Attributes

- `McsController * controller`
- `Satellite * sat`

Constructor & Destructor Documentation

`acknowledged (Satellite * tmpSat, McsController * tmpController)` [inline]

Member Function Documentation

`void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value *const retvalP)` [inline]

Member Data Documentation

`McsController* controller` [private]

`Satellite* sat` [private]

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

- `server/mcsListener.cpp`

3.2.14 AdminLowlevel Class Reference

Class that presents an administration interface to the lower layers of the system.

`#include <adminlowlevel.h>`

Inherited by `DatabaseAbstraction`. 
Inheritance diagram for AdminLowlevel:

```
AdminLowlevel
 + accountAddExperiment()
 + accountAddPayload()
 + accountExists()
 + accountHasRightsToExperiment()
 + accountHasRightsToPayload()
 + accountHasRightsToSatellite()
 + accountRemoveExperiment()
 + accountRemovePayload()
 + addLog()
 + getAllBookingIDs()
 + getAllCommandIDs()
 + getAllDataIDs()
 + getAllExperimentIDs()
 + getAllPayloadIDs()
 + getAllSatelliteIDs()
 + getAvailableSatellites()
 + getList()
 + getExperimentAtTime()
 + getExternalBookingsOfExperiment()
 + getFirstBooking()
 + getInternalBooking()
 + getPayloadExperiments()
 + getPayloadListOfCommands()
 + getQueue()
 + getThumbnailList()
 + getTimeBookingIDs()
 - removeExperiment()
 - removeQueue()
 - removeQueueItem()
 - retrieveAccount()
 - retrieveBooking()
 - retrieveCommand()
 - retrieveData()
 - retrieveExperiment()
 - retrieveLog()
 - retrieveLogs()
 - retrievePayload()
 - retrieveQueueItem()
 - retrieveSatellite()
 - updateAccount()
 - updateBooking()
 - updateCommand()
 - updateData()
 - updateExperiment()
 - updatePayload()
 - updateQueueItem()
 - updateSatellite()
 ~AdminLowlevel()
```

DatabaseAbstraction

```
connection
db
host
passwd
user
numberOfRetries
 + accountAddExperiment()
 + accountAddPayload()
 + accountExists()
 + accountHasRightsToExperiment()
 + accountHasRightsToPayload()
 + accountHasRightsToSatellite()
 + accountRemoveExperiment()
 + accountRemovePayload()
 + addLog()
 + connect()
 + DatabaseAbstraction()
 + getAllBookingIDs()
 + getAllCommandIDs()
 + getAllDataIDs()
 + getAllExperimentIDs()
 + getAllPayloadIDs()
 + getAllSatelliteIDs()
 + getAvailableSatellites()
 + getList()
 + getExperimentAtTime()
 + getExternalBookingsOfExperiment()
 + getFirstBooking()
 + getInternalBooking()
 + getPayloadExperiments()
 + getPayloadListOfCommands()
 + getQueue()
 + getThumbnailList()
 + getTimeBookingIDs()
 - removeExperiment()
 - removeQueue()
 - removeQueueItem()
 - retrieveAccount()
 - retrieveBooking()
 - retrieveCommand()
 - retrieveData()
 - retrieveExperiment()
 - retrieveLog()
 - retrieveLogs()
 - retrievePayload()
 - retrieveQueueItem()
 - retrieveSatellite()
 - updateAccount()
 - updateBooking()
 - updateCommand()
 - updateData()
 - updateExperiment()
 - updatePayload()
 - updateQueueItem()
 - updateSatellite()
 ~DatabaseAbstraction()
```

DDD 0.1


CHAPTER 3. COMPONENT DESCRIPTIONS

Public Member Functions

- virtual void accountAddExperiment (const AccountID &aid, const ExperimentID &eid) const=0
  
  Gives the account access to an experiment.

- virtual void accountAddPayload (const AccountID &aid, const PayloadID &pid) const=0
  
  Give this account access to a payload.

- virtual bool accountExists (AccountID id) const =0
  
  Checks whether the current account exists in the database.

- virtual bool accountHasRightsToExperiment (const AccountID &aid, const ExperimentID &eid) const=0
  
  Returns if the account with identifier aid has rights to access the experiment with identifier eid.

- virtual bool accountHasRightsToPayload (const AccountID &aid, const PayloadID &pid) const=0
  
  Returns if the account with identifier aid has rights to access the payload with identifier pid.

- virtual bool accountHasRightsToSatellite (const AccountID &aid, const SatelliteID &sid) const =0
  
  Returns whether this account has access to the satellite with satelliteID.

- virtual void accountRemoveExperiment (const AccountID &aid, const ExperimentID &eid) const=0
  
  Removes the rights to access an experiment.

- virtual void accountRemovePayload (const AccountID &aid, const PayloadID &pid) const=0
  
  Remove the rights to access to a payload.

- virtual void addLog (SingleLog aLog) const=0
  
  Add a new log to the database.

- virtual std::vector< BookingID > getAllBookingIDs (BookingType &type) const=0
  
  Gets the IDs of all bookings in the database.

- virtual std::vector< CommandID > getAllCommandIDs () const=0
  
  Gets the IDs of all satellites in the database.

- virtual std::vector< DataID > getAllDataIDs () const=0
  
  Gets the IDs of all data items in the database.

- virtual std::vector< ExperimentID > getAllExperimentIDs () const=0
  
  Gets the IDs of all experiments in the database.

- virtual std::vector< PayloadID > getAllPayloadIDs () const=0
  
  Gets the IDs of all payloads in the database.
• virtual std::vector< SatelliteID > getAllSatelliteIDs () const=0
  Gets the IDs of all satellites in the database.

• virtual std::vector< SatelliteID > getAvailableSatellites (int time) const=0
  Retrieve all satellites that have a pass at a given time.

• virtual DataList getDataList (const ExperimentID &id) const=0
  Retrieves the list of data items belonging to the experiment with identifier id.

• virtual ExperimentID getExperimentAtTime (const SatelliteID &satelliteID, const unsigned int time) const=0
  Retrieve the experiment at a specific time for a specific satellite.

• virtual std::vector< BookingID > getExternalBookingsOfExperiment (ExperimentID id) const=0
  get the associated external bookings of an experiment.

• virtual BookingID getFirstBooking () const=0
  Gets the first booking from now.

• virtual BookingID getInternalBooking (SatelliteID satelliteID) const=0
  Get the bookingId for the satellite, return -1 if not exists.

• virtual std::vector< ExperimentID > getPayloadExperiments (const PayloadID &id) const=0
  Retrieves the experiment identifier of the experiments defined on the payload.

• virtual std::vector< CommandID > getPayloadListOfCommands (const PayloadID &id) const=0
  Retrieves identifier of all commands applicable to the payload with identifier id.

• virtual std::list< QueueItem > getQueue (const SatelliteID &satelliteID, const BookingType &type) const=0
  Retrieve all queueItems belonging to the queue with identifier (satelliteID, type).

• virtual DataList getThumbnailList (const ExperimentID &id) const=0
  Retrieves the list of thumbnails belonging to the experiment with identifier id.

• virtual std::vector< BookingID > getTimeBookingIDs (int time) const=0
  Retreive all bookings for a given time.

• virtual void removeExperiment (const ExperimentID &id) const=0
  Removes the experiment with identifier id.

• virtual void removeQueue (const SatelliteID &satelliteID, const BookingType &type) const=0
  Remove all queueItems belonging to the queue with identifier, where history == false. (satelliteID, type).

• virtual void removeQueueItem (const ItemID &id) const=0
CHAPTER 3. COMPONENT DESCRIPTIONS

Removes queueitem with itemid id.

- virtual void retrieveAccount (const AccountID &id, AccountType &type, std::string &password, bool &registered, std::string &fullname, std::string &email) const=0
  Retrieves the account information from the account with identifier id.

- virtual void retrieveBooking (const BookingID &id, Direction &direction, BookingType &type, BookingStatus &status, int &startTime, int &endTime, ExperimentID &experimentID, GSSID &gssID) const=0
  Retrieves the booking information from the booking with identifier id.

- virtual void retrieveCommand (const CommandID &id, std::string &name, PayloadID &payload, CommandParam &commandParam) const=0
  Retrieves the payload information from the payload with identifier id.

- virtual void retrieveData (const DataID &id, std::string &name, ExperimentID &experiment, int &gatheredAt, std::string &type, char **dataItem, int &dataItemLength, char **thumbnail, int &thumbnailLength, bool &markedForDeletion) const=0
  Retrieves the data information from the data with identifier id.

- virtual void retrieveExperiment (const ExperimentID &id, std::string &name, PayloadID &payload, int &scheduledFrom, int &scheduledTo, ExperimentStatus &status, ListOfExperimentCommands &listOfCommands, ListOfCommands &listOfPossibleCommands) const=0
  Retrieves the experiment information from the experiment with identifier id.

- virtual SingleLog retrieveLog (LogID id) const=0
  retrieve the log with a certain identifier.

- virtual std::vector<LogID> retrieveLogs (int begintime, int endtime, const SingleLog::Component &source, int priority, const ExperimentID &anExperimentID, const PayloadID &aPayloadID, const SatelliteID &aSatelliteID) const=0
  return a list of LogIDs that are between begintime and endtime, are from component source and have a priority priority.

- virtual void retrievePayload (const PayloadID &id, std::string &name, bool &registered, SatelliteID &satelliteID) const=0
  Retrieves the payload information from the payload with identifier id.

- virtual void retrieveQueueItem (const ItemID &id, Action &action, SatelliteID &satID, PayloadID &payloadID, int &intVal, std::string &strVal, BookingType &type, bool &history) const=0
  Retrieves the queue information from the queue with identifier the tuple (satelliteID, type).

- virtual void retrieveSatellite (const SatelliteID &id, std::string &name, bool &registered, std::string &mccUrl, std::string &mcsUrl) const=0
  Retrieves the payload information from the payload with identifier id.

- virtual void updateAccount (AccountID &aid, const AccountType &type, const std::string &password, const bool registered, const std::string &fullname, const std::string &email) const=0
CHAPTER 3. COMPONENT DESCRIPTIONS

Updates the account information from the account with identifier id.

- virtual void updateBooking (BookingID &id, const Direction &direction, const BookingType &type, const BookingStatus &status, const int startTime, const int endTime, const ExperimentID &experimentID, const GSSID &gssID) const=0

  Updates the booking information from the booking with identifier id.

- virtual void updateCommand (CommandID &id, const std::string &name, const PayloadID &payload, const CommandParam &commandParam) const=0

  Updates the command information from the command with identifier id.

- virtual void updateData (DataID &id, const std::string &name, const ExperimentID &experiment, const int &gatheredAt, const std::string &type, const char *dataItem, const int &dataItemLength, const char *thumbnail, const int &thumbnailLength, const bool markedForDeletion) const=0

  Updates the data information from the data with identifier id.

- virtual void updateExperiment (ExperimentID &id, const std::string &name, const PayloadID &payload, const int &scheduledFrom, const int &scheduledTo, const ExperimentStatus &status, const ListOfExperimentCommands &listOfPossibleCommands) const=0

  Updates the experiment information from the experiment with identifier id.

- virtual void updatePayload (PayloadID &id, const std::string &name, const bool registered, const SatelliteID &satelliteID) const=0

  Updates the payload information from the payload with identifier id.

- virtual void updateQueueItem (ItemID &id, const Action &action, const SatelliteID &satID, const PayloadID &payloadID, const int intVal, const std::string &strVal, const BookingType &type, const bool history) const=0

  Updates the queueitem information from the queueitem with identifier id.

- virtual void updateSatellite (SatelliteID &id, const std::string &name, const bool registered, const std::string &mccUrl, const std::string &mcsUrl) const=0

  Updates the satellite information from the satellite with identifier id.

- virtual ~AdminLowlevel ()

  Destructor of the AdminLowlevel (p. 126) class.

Classes

- class AdminLowlevelException

  Exception class thrown by AdminLowlevel (p. 126).

Constructor & Destructor Documentation

virtual ~AdminLowlevel () [inline, virtual]

Precondition:

  true
Member Function Documentation

virtual void accountAddExperiment (const AccountID & aid, const ExperimentID & eid) const  [pure virtual]

Parameters:
    aid  The identifier of the account.
    eid  The identifier of the experiment.

Precondition:
    account.registered = true

Postcondition:
    account.hasRightsToExperiment(eID) = true

Implemented in DatabaseAbstraction  (p. 215).

virtual void accountAddPayload (const AccountID & aid, const PayloadID & pid)
    const  [pure virtual]

Parameters:
    aid  The identifier of the account.
    pid  The identifier of the payload.

Precondition:
    account.registered = true

Postcondition:
    account.hasRightsToPayload(aid, pid) and (Forall e in Experiment (p. 221) (e.getPayload() = pid => account.hasRightsToExperiment(e)))

Implemented in DatabaseAbstraction  (p. 215).

virtual bool accountExists (AccountID id) const  [pure virtual]

Parameters:
    id  The Account (p. 114) identifier

Precondition:
    true

Returns:
    (exists a in Account (p. 114) ( a.identifier = this.getID))

Implemented in DatabaseAbstraction  (p. 215).
virtual bool accountHasRightsToExperiment (const AccountID & aid, const ExperimentID & eid) const  [pure virtual]

Parameters:
  
  *aid* The identifier of the account.
  
  *eid* The identifier of the experiment.

Precondition:
  
  true

Returns:
  
  Exists an eID’ in Experiment (p. 221):: eID’ = eid and this.canObserve = eid \ ( account(aid).getType () == Scientist \ account(aid).hasRightsToPayload(experiment(eid).getPayload (p. 240)) )

Implemented in DatabaseAbstraction (p. 215).

virtual bool accountHasRightsToPayload (const AccountID & aid, const PayloadID & pid) const  [pure virtual]

Parameters:
  
  *aid* The identifier of the account.
  
  *pid* The identifier of the payload.

Precondition:
  
  true

Returns:
  
  Exists a pID’ in Payload (p. 276):: pID’ = id and this.hasAccessTo = id.

Implemented in DatabaseAbstraction (p. 215).

virtual bool accountHasRightsToSatellite (const AccountID & aid, const SatelliteID & sid) const  [pure virtual]

Parameters:
  
  *aid* The identifier of the account.
  
  *sid* a satellite identifier

Precondition:
  
  true

Returns:
  
  Whether this user has access to a Payload (p. 276) or an Experiment (p. 221) defined on this satellite.

Implemented in DatabaseAbstraction (p. 215).
virtual void accountRemoveExperiment (const AccountID & aid, const ExperimentID & eid) const  [pure virtual]

Parameters:
  
  aid The identifier of the account.
  
  eid The identifier of the experiment.

Precondition:
  
  account.registered = true

Postcondition:
  
  account.hasRightsToExperiment(eID) = false

Implemented in DatabaseAbstraction (p. 215).

virtual void accountRemovePayload (const AccountID & aid, const PayloadID & pid) const  [pure virtual]

Parameters:
  
  aid The identifier of the account.
  
  pid The identifier of the payload.

Precondition:
  
  account.registered = true

Postcondition:
  
  account.hasRightToPayload(pID) = false and (Forall e in Experiment (p. 221) (e.getPayload() = pid => not(account.hasRightsToExperiment(e))))

Implemented in DatabaseAbstraction (p. 215).

virtual void addLog (SingleLog aLog) const  [pure virtual]

Parameters:
  
  aLog the log to be added.

Precondition:
  
  true

Postcondition:
  
  Log added to logs in the database.

Implemented in DatabaseAbstraction (p. 215).
virtual std::vector<BookingID> getAllBookingIDs (BookingType & type) const  [pure virtual]

Precondition:
true
Postcondition:
{ b.Id | b in connection.Booking /\ b.type = type }

Implemented in DatabaseAbstraction  (p. 216).

virtual std::vector<CommandID> getAllCommandIDs () const  [pure virtual]

Precondition:
true
Postcondition:
{ s.Id | s in Satellite (p. 293) }

Implemented in DatabaseAbstraction  (p. 216).

virtual std::vector<DataID> getAllDataIDs () const  [pure virtual]

Precondition:
true
Postcondition:
{ d.Id | d in Data (p. 199) }

Implemented in DatabaseAbstraction  (p. 216).

virtual std::vector<ExperimentID> getAllExperimentIDs () const  [pure virtual]

Precondition:
true
Postcondition:
{ e.Id | e in Experiment (p. 221) }

Implemented in DatabaseAbstraction  (p. 216).

virtual std::vector<PayloadID> getAllPayloadIDs () const  [pure virtual]

Precondition:
true
Postcondition:
{ p.Id | p in Payload (p. 276) }

Implemented in DatabaseAbstraction  (p. 216).
virtual std::vector<SatelliteID> getAllSatelliteIDs () const

Precondition:
true

Postcondition:
{ s.Id | s in Satellite (p. 293) }

Implemented in DatabaseAbstraction (p. 216).

virtual std::vector<SatelliteID> getAvailableSatellites (int time) const

Parameters:

time Time for which the satellites have to be retrieved.

Returns:
All satelliteID's of the satellite that have a pass at the given time and are registered.

Implemented in DatabaseAbstraction (p. 216).

virtual DataList getDataList (const ExperimentID & id) const

Parameters:

id an experimentID

Precondition:

id >= 0

Returns:
the list of data items belonging to experiment with identifier id.

Implemented in DatabaseAbstraction (p. 216).

virtual ExperimentID getExperimentAtTime (const SatelliteID & satelliteID, const unsigned int time) const

Parameters:

satelliteID Representing a satellite.
time A time.

Precondition:
true

Returns:
An experimentID e: e.scheduledFrom <= time <= e.scheduledTo, else -1

Implemented in DatabaseAbstraction (p. 216).
virtual std::vector<BookingID> getExternalBookingsOfExperiment (ExperimentID id) const  [pure virtual]

Parameters:

    id  The experimentid for which the bookings are to be fetched.

Precondition:

    true

Returns:

    \{b.id | b in connection.Booking /\ b.getExperimentID = id /\ b.type = External\}

Implemented in DatabaseAbstraction  (p. 217).

virtual BookingID getFirstBooking () const  [pure virtual]

Precondition:

    true

Returns:

    b.identifier and Forall (b' in Booking (p. 150) /\ b'.status = Accepted :: b'.starttime >=
    b.starttime). Returns -1 if there is no booking in the database.

Implemented in DatabaseAbstraction  (p. 217).

virtual BookingID getInternalBooking (SatelliteID satelliteID) const  [pure virtual]

Parameters:

    satelliteID  The satelliteID for which we want the internal booking.

Precondition:

    true

Returns:

    BookingID of internal booking with status either "created" or "pending" or "accepted" if the
    internal booking for satellite exists else -1.

Implemented in DatabaseAbstraction  (p. 217).

virtual std::vector<ExperimentID> getPayloadExperiments (const PayloadID & id) const  [pure virtual]

Parameters:

    id  The identifier of the payload.
Precondition:
  true
Returns:
  \{e.identifier | (e in Experiment \text{ (p. 221)}) \land (e.payloadID = id)\}
Implemented in DatabaseAbstraction (p. 217).

```cpp
virtual std::vector<CommandID> getPayloadListOfCommands (const PayloadID & id) const [pure virtual]
```

Parameters:
- `id` The identifier of the payload.

Precondition:
  true
Returns:
  \{c.identifier | (c in Command \text{ (p. 162)}) \land (c.payloadID = id)\}
Implemented in DatabaseAbstraction (p. 217).

```cpp
virtual std::list<QueueItem> getQueue (const SatelliteID & satelliteID, const BookingType & type) const [pure virtual]
```

Parameters:
- `satelliteID` A satellite to which the queue belongs to.
- `type` The bookingtype of the queue.

Precondition:
  true
Returns:
  A list of queueItems belonging to (satelliteID, type). If satelliteID == -1, return empty list.
Implemented in DatabaseAbstraction (p. 217).

```cpp
virtual DataList getThumbnailList (const ExperimentID & id) const [pure virtual]
```

Parameters:
- `id` an experimentID

Precondition:
  id \geq 0
Returns:
  the list of thumbnails belonging to experiment with identifier id.
Implemented in DatabaseAbstraction (p. 217).
virtual std::vector<BookingID> getTimeBookingIDs (int time) const [pure virtual]

Parameters:
  \textit{time} Time for which the bookings have to be retrieved.

Returns:
  All bookings for during a given time.

Implemented in \texttt{DatabaseAbstraction} (p. 217).

virtual void removeExperiment (const ExperimentID & \textit{id}) const [pure virtual]

Parameters:
  \textit{id} an experimentID

Precondition:
  id >= 0

Returns:
  not (exists e in experiment: e.id = id)

Implemented in \texttt{DatabaseAbstraction} (p. 217).

virtual void removeQueue (const SatelliteID & \textit{satelliteID}, const BookingType & \textit{type}) const [pure virtual]

Parameters:
  \textit{satelliteID} A satellite to which the queue belongs to.
  \textit{type} The bookingtype of the queue.

Precondition:
  true

Postcondition:
  The list of queueItems belonging to (satelliteID, type) is removed. If satelliteID == -1, skip.

Implemented in \texttt{DatabaseAbstraction} (p. 218).

virtual void removeQueueItem (const ItemID & \textit{id}) const [pure virtual]

Parameters:
  \textit{id} The itemid to be removed.

Precondition:
  true
Postcondition:

id not in queue.

Implemented in DatabaseAbstraction (p. 218).

virtual void retrieveAccount (const AccountID & id, AccountType & type, std::string & password, bool & registered, std::string & fullname, std::string & email) const [pure virtual]

Parameters:

id The identifier of the account.

type The type of the account.

password the password to access the account.

registered Whether the account is registered.

fullname The fullname of the user.

email The email adress of the user.

Precondition:

ture

Postcondition:

account.id, account.type, account.password, account.registered, account.fullname and account.email are retrieved.

Implemented in DatabaseAbstraction (p. 218).

virtual void retrieveBooking (const BookingID & id, Direction & direction, BookingType & type, BookingStatus & status, int & startTime, int & endTime, ExperimentID & experimentID, GSSID & gssID) const [pure virtual]

Parameters:

id The identifier of the booking.

direction Direction of the communication between the MCS and the satellite.

type Whether the booking is internal or external.

status Current status of the booking.

startTime The starttime of the booking.

endTime The endtime of the booking

experimentID -1 if the booking is an internal booking, otherwise the experiment for which the external booking was made.

gssID The groundstation of the booking.

Precondition:

ture
CHAPTER 3. COMPONENT DESCRIPTIONS

Postcondition:

booking.direction = direction /
booking.type = type /
booking.status = status /
booking.startTime = startTime /
booking.endTime = endTime /
booking.experimentID = experimentID /
booking.gssID = gssid

Implemented in DatabaseAbstraction (p. 218).

virtual void retrieveCommand (const CommandID & id, std::string & name, PayloadID & payload, CommandParam & commandParam) const  [pure virtual]

Parameters:

id The identifier of the command.
name The name of the command.
payload The payload to which the command belongs to.
commandParam The parameters of the command.

Precondition:

true

Postcondition:

command.name, command.payload and command.commandParam are retrieved.

Implemented in DatabaseAbstraction (p. 218).

virtual void retrieveData (const DataID & id, std::string & name, ExperimentID & experiment, int & gatheredAt, std::string & type, char ** dataItem, int & dataItemLength, char ** thumbnail, int & thumbnailLength, bool & markedForDeletion) const  [pure virtual]

Parameters:

id The identifier of the data.
name The name of the data.
experiment The experiment to which the data belongs to.
gatheredAt The time at which the data is gathered.
type The datatype of the data.
dataItem The data item, which can be a measurement or a high-res image.
dataItemLength The length of the data item.
thumbnail A thumbnail.
thumbnailLength The length of the thumbnail.
markedForDeletion Whether the data is marked for deletion.

Precondition:

true

Postcondition:

data.name, data.experiment, data.gatheredAt, data.type, data.dataItem, data.dataItemLength, data.thumbnail, data.thumbnailLength, data.markedForDeletion are retrieved

Implemented in DatabaseAbstraction (p. 218).
virtual void retrieveExperiment (const ExperimentID & id, std::string & name, PayloadID & payload, int & scheduledFrom, int & scheduledTo, ExperimentStatus & status, ListOfExperimentCommands & listOfCommands, ListOfCommands & listOfPossibleCommands) const [pure virtual]

Parameters:

  id The identifier of the experiment.
  name The name of the experiment.
  payload The PayloadID the experiment belongs to.
  scheduledFrom Start time of the schedule.
  scheduledTo End time of the schedule.
  status The status of the experiment.
  listOfCommands The list of commands the experiment is composed of.
  listOfPossibleCommands The list of possible commands of the payload to which the experiment belongs to.

Precondition:
true

Postcondition:
experiment parameters are retrieved

Implemented in DatabaseAbstraction (p. 218).

virtual SingleLog retrieveLog (LogID id) const [pure virtual]

Parameters:

  id The identifier of a certain log to be retrieved.

Precondition:
LogID > -1

Postcondition:
SingleLog.ID = LogID

Implemented in DatabaseAbstraction (p. 218).

virtual std::vector<LogID> retrieveLogs (int begintime, int endtime, const SingleLog:: Component & source, int priority, const ExperimentID & anExperimentID, const PayloadID & aPayloadID, const SatelliteID & aSatelliteID) const [pure virtual]

Parameters:

  begintime get logs starting from time begintime.
  endtime get logs ending at time endtime.
  source get logs which were made by the component source.
priority get logs with priority priority.
anExperimentID get logs with experimentID anExperimentID.
aPayloadID get logs with payloadID aPayloadID.
aSatelliteID get logs with satelliteID satelliteID.

Precondition:
true

Postcondition:
All logIDs passing the filter.

Implemented in DatabaseAbstraction (p. 218).

virtual void retrievePayload (const PayloadID & id, std::string & name, bool & registered, SatelliteID & satelliteID) const [pure virtual]

Parameters:
id The identifier of the payload.
name The name of the payload.
registered Whether the payload is registered.
satelliteID Identifier of the satellite to which the payload is linked.

Precondition:
true

Postcondition:
payload.name, payload.registered and payload.satelliteID are retrieved.

Implemented in DatabaseAbstraction (p. 219).

virtual void retrieveQueueItem (const ItemID & id, Action & action, SatelliteID & satID, PayloadID & payloadID, int & intVal, std::string & strVal, BookingType & type, bool & history) const [pure virtual]

Parameters:
id An ItemID identifying the queueItem.
action An action.
satID A satelliteID representing the satellite the queue item belongs to.
payloadID A payloadID representing the payload the queue item belongs to.
intVal A integer value (optional).
strVal A string value (optional).
type The bookingtype of the queueitem.
history Whether the queueitem belongs to the history or not.
CHAPTER 3. COMPONENT DESCRIPTIONS

Precondition:
true

Postcondition:
queueitem.itemID, queueitem.action, queueitem.satID, queueitem.payloadID, queueitem.intVal,
queueitem.strVal, queueitem.type, queueitem.history are retrieved

Implemented in DatabaseAbstraction (p. 219).

virtual void retrieveSatellite (const SatelliteID & id, std::string & name, bool & registered, std::string & mccUrl, std::string & mcsUrl) const [pure virtual]

Parameters:
  id The identifier of the satellite.
  name The name of the satellite.
  registered Whether the satellite is registered.
  mccUrl Location of the MCC for this satellite.
  mcsUrl Location of the MCS for this satellite.

Precondition:
true

Postcondition:
satellite.name, satellite.registered, satellite.mccUrl and satellite.mcsUrl are retrieved.

Implemented in DatabaseAbstraction (p. 219).

virtual void updateAccount (AccountID & aid, const AccountType & type, const std::string & password, const bool registered, const std::string & fullname, const std::string & email) const [pure virtual]

Parameters:
  aid The identifier of the account.
  type The type of the account.
  password The password to access the account.
  registered Whether the account is registered.
  fullname The fullname of the user.
  email The email adress of the user.

Precondition:
true

Postcondition:
account.id = id /\ account.type = type /\ account.password = password /\ account.registered
= registered /\ account.fullname = fullname /\ account.email = email (if id="", then a new
account object is created and the account identifier is locally updated).

Implemented in DatabaseAbstraction (p. 219).
virtual void updateBooking (BookingID & id, const Direction & direction, const BookingType & type, const BookingStatus & status, const int startTime, const int endTime, const ExperimentID & experimentID, const GSSID & gssID) const [pure virtual]

Parameters:

id The identifier of the booking.
direction Direction of the communication between the MCS and the satellite.
type Whether the booking is internal or external.
status Current status of the booking.
startTime The start time of the booking.
endTime The end time of the booking.
experimentID -1 if the booking is an internal booking, otherwise the experiment for which the external booking was made.
gssID The ground station of the booking.

Precondition:

true

Postcondition:

booking.direction = direction \ booking.type = type \ booking.status = status \ booking.startTime = startTime \ booking.endTime = endTime \ booking.experimentID = experimentID \ booking.gssID = gssid (if id=-1, then a new booking object is created and the booking identifier is locally updated)

Implemented in DatabaseAbstraction (p. 219).

virtual void updateCommand (CommandID & id, const std::string & name, const PayloadID & payload, const CommandParam & commandParam) const [pure virtual]

Parameters:

id The identifier of the command.
name The name of the command.
payload The payload to which the command belongs to.
commandParam The parameters of the command.

Precondition:

true

Postcondition:

command.name = name \ command.payload = payload \ command.commandParam = commandParam (if id=-1, then a new command object is created and the command identifier is locally updated)

Implemented in DatabaseAbstraction (p. 219).
virtual void updateData (DataID & id, const std::string & name, const ExperimentID & experiment, const int & gatheredAt, const std::string & type, const char * dataItem, const int & dataItemLength, const char * thumbnail, const int & thumbnailLength, const bool markedForDeletion) const [pure virtual]

Parameters:

  id The identifier of the data.
  name The name of the data.
  experiment The experiment to which the data belongs to.
  gatheredAt The time at which the data is gathered.
  type The datatype of the data.
  dataItem The data item, which can be a measurement or a high-res image.
  dataItemLength The length of the data item.
  thumbnail A thumbnail.
  thumbnailLength The length of the thumbnail.
  markedForDeletion Whether the data is marked for deletion.

Precondition:

  true

Postcondition:

  data.name = name /
  data.experiment = experiment /
  data.gatheredAt = gatheredAt /
  data.type = type /
  data.dataItem = dataItem /
  data.dataItemLength = dataItemLength /
  data.thumbnail = thumbnail /
  data.thumbnailLength = thumbnailLength /
  data.markedForDeletion = markedForDeletion

Implemented in DatabaseAbstraction (p. 219).

virtual void updateExperiment (ExperimentID & id, const std::string & name, const PayloadID & payload, const int & scheduledFrom, const int & scheduledTo, const ExperimentStatus & status, constListOfExperimentCommands & listOfPossibleCommands) const [pure virtual]

Parameters:

  id The identifier of the experiment.
  name The name of the experiment.
  payload The PayloadID the experiment belongs to.
  scheduledFrom Start time of the schedule.
  scheduledTo End time of the schedule.
  status The status of the experiment.
  listOfPossibleCommands The list of commands the experiment is composed of.

Precondition:

  true

Postcondition:

  experiment parameters are updated.

Implemented in DatabaseAbstraction (p. 220).
virtual void updatePayload (PayloadID & id, const std::string & name, const bool registered, const SatelliteID & satelliteID) const [pure virtual]

Parameters:
  id The identifier of the payload.
  name The name of the payload.
  registered Whether the payload is registered.
  satelliteID Identifier of the satellite to which the payload is linked.

Precondition:
  true

Postcondition:
  payload.name = name /\ payload.registered = registered /\ payload.satelliteID = satelliteID
  (if id=-1, then a new payload object is created and the payload identifier is locally updated)

Implemented in DatabaseAbstraction  (p. 220).

virtual void updateQueueItem (ItemID & id, const Action & action, const SatelliteID & satID, const PayloadID & payloadID, const int intVal, const std::string & strVal, const BookingType & type, const bool history) const [pure virtual]

Parameters:
  id An ItemID identifying the queueItem.
  action An action.
  satID A satelliteID representing the satellite the queue item belongs to.
  payloadID A payloadID representing the payload the queue item belongs to.
  intVal A integer value (optional).
  strVal A string value (optional).
  type The bookingtype of the queueitem.
  history Whether the queueitem belongs to the history or not.

Precondition:
  true

Postcondition:
  parameters of queueItem are updated.

Implemented in DatabaseAbstraction  (p. 220).

virtual void updateSatellite (SatelliteID & id, const std::string & name, const bool registered, const std::string & mccUrl, const std::string & mcsUrl) const [pure virtual]

Parameters:
  id The identifier of the satellite.
** CHAPTER 3. COMPONENT DESCRIPTIONS **

- **name** The name of the satellite.
- **registered** Whether the satellite is registered.
- **mccUrl** Location of the MCC for this satellite.
- **mcsUrl** Location of the MCS for this satellite.

**Precondition:**
true

**Postcondition:**
satellite.name = name /
registered = registered /
mccUrl = mceUrl /
mcsUrl = mcsUrl (if id=-1, then a new satellite object is created and the satellite identifier is locally updated)

Implemented in **DatabaseAbstraction** (p. 220).
The documentation for this class was generated from the following file:
- common/adminlowlevel.h

### 3.2.15 AdminLowlevel::AdminLowlevelException Class Reference

Exception class thrown by **AdminLowlevel** (p. 126).

- **include <adminlowlevel.h>**

Inherits **runtime_error**.
Inherited by **DatabaseAbstraction::DatabaseException**.

Inheritance diagram for AdminLowlevel::AdminLowlevelException:

```
    std::runtime_error
    + AdminLowlevelException()

    AdminLowlevel::AdminLowlevelException
    + AdminLowlevelException()

    DatabaseException
    + DatabaseException()
```

**Public Member Functions**
- **AdminLowlevelException** (const std::string &arg)

**Constructor & Destructor Documentation**

AdminLowlevelException (const std::string & arg) [inline]
CHAPTER 3. COMPONENT DESCRIPTIONS

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

- common/adminlowlevel.h

3.2.16 booked Class Reference

Inherits xmlrpc_c::method.

Inheritance diagram for booked:

```
        method
         |
        booked
         |  - controller
         |  - sat
         |  + booked()
         |  + execute()
```

Public Member Functions

- booked (Satellite *tmpSat, MccController *tmpController)
- void execute (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)

Private Attributes

- MccController * controller
- Satellite * sat

Constructor & Destructor Documentation

booked (Satellite * tmpSat, MccController * tmpController) [inline]

Member Function Documentation

void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value * const retvalP) [inline]

Member Data Documentation

MccController* controller [private]
Satellite* sat  [private]
The documentation for this class was generated from the following file:

- server/mccListener.cpp

### 3.2.17 Booking Class Reference

```cpp
#include <booking.h>
```

**Public Member Functions**

- **Booking** (const BookingID id)
  
  Constructor of a new booking object with BookingID id.

- **Booking** (const Direction direction, const BookingType type, const BookingStatus status, const int startTime, const int endTime, const ExperimentID experimentID, const GSSID gssid)
  
  Constructor of a new booking object.

- **Booking** ()
  
  Empty constructor.

- **Direction** getDirection ()
  
  Returns the direction of the communication between the MCS and the satellite.

- **int** getEndTime ()
  
  Returns the endtime of the booking.

- **ExperimentID** getExperimentID ()
  
  Returns the experimentID to which the booking belongs.

- **GSSID** getGssID ()
  
  Returns the groundstation of the booking.

- **BookingID** getID () const
  
  Returns the identifier for this booking.

- **int** getStartTime ()
  
  Returns the starttime of the booking.

- **BookingStatus** getStatus ()
  
  Returns the status of the booking.

- **BookingType** getType ()
  
  Returns the type of the booking.

- **void** instantiate ()
  
  Retrieves data from booking with PayloadID from lower layer.
void setDirection (Direction direction)
    
    Sets the direction of the communication between the MCS and the satellite.

void setEndTime (int endTime)
    
    Sets the endtime of the booking.

void setExperimentID (ExperimentID id)
    
    Sets the experimentID to which the booking belongs in case the booking is an external booking.

void setGssID (const GSSID &gssid)
    
    Sets the gssid.

void setStartTime (int startTime)
    
    Sets the starttime of the booking.

void setStatus (BookingStatus status)
    
    Sets the status of the booking.

void setType (BookingType type)
    
    Sets the type of the booking.

void store ()
    
    Stores the data of the booking into the database.

~Booking ()
    
    Destructor of a booking object.

Private Attributes

- Direction direction
- int endTime
    
    The endtime of the booking.

- ExperimentID experimentID
- GSSID gssid
    
    Identifying the groundstation the booking belongs to.

- BookingID identifier
    
    Identifier for this object in the lower layer.

- int startTime
    
    The starttime of the booking.

- BookingStatus status
    
    Current status of the booking.

- BookingType type
    
    Whether the booking is internal or external.
Classes

- class BookingException

Constructor & Destructor Documentation

Booking ()

Precondition:
true

Postcondition:
this.identifier = -1 \ this.direction = "TwoWay" \ this.type = "Internal" \ this.status = "Created" \ this.startTime = -1 \ this.endTime = -1 \ this.experimentID = -1

Booking (const Direction direction, const BookingType type, const BookingStatus status, const int startTime, const int endTime, const ExperimentID experimentID, const GSSID gssid)

Parameters:
direction Direction of the communication between the MCS and the satellite.
type Whether the booking is internal or external
status Current status of the booking
startTime The start time of the booking
dTime The end time of the booking
experimentID -1 if the booking is an internal booking, otherwise the experiment for which the external booking was made.
gssid The gssid belonging to the booking.

Precondition:
experimentID > -1

Postcondition:
this.identifier = unique identifier \ this.direction = direction \ this.type = type \ this.status = status \ this.startTime = startTime \ this.endTime = endTime \ this.experimentID = experimentID

Booking (const BookingID id)

Parameters:
id a BookingID identifying the booking in the lower layer.

Precondition:
id >= 0
CHAPTER 3. COMPONENT DESCRIPTIONS

Postcondition:
\[
\text{this.identifier} = \text{id} /\langle\text{this.direction} = \text{direction} /\langle\text{this.type} = \text{type} /\langle\text{this.status} = \text{status} \\
/\langle\text{this.startTime} = \text{startTime} /\langle\text{this.endTime} = \text{endTime} /\langle\text{this.experimentID} = \text{experimentID}
\]

~Booking ()

Precondition:
false

Member Function Documentation

Direction getDirection ()

Precondition:
false

Postcondition:
this.direction

int getEndTime ()

Precondition:
false

Returns:
this.endTime

ExperimentID getExperimentID ()

Precondition:
false

Returns:
this.experimentID

GSSID getGssID ()

Precondition:
\text{id} > -1

Returns:
this.gssID
BookingID getID () const

Precondition:
    true

Returns:
    this.identifier

int getStartTime ()

Precondition:
    true

Returns:
    this.startTime

BookingStatus getStatus ()

Precondition:
    true

Returns:
    this.status

BookingType getType ()

Precondition:
    true

Returns:
    this.type

void instantiate ()

Precondition:
    this.getID() >= 0

Postcondition:
    The parameters of the booking are instantiated with information from the lower layer.
void setDirection (Direction \textit{direction})

Precondition:
\texttt{true}

Postcondition:
\texttt{this.direction = direction}

void setEndTime (int \textit{endTime})

Precondition:
\texttt{true}

Postcondition:
\texttt{this.endTime = endTime}

void setExperimentID (ExperimentID \textit{id})

Precondition:
\texttt{id > -1}

Postcondition:
\texttt{this.experimentID = id}

void setGssID (const GSSID & \textit{gssid})

Precondition:
\texttt{true}

Postcondition:
\texttt{this.gssID = gssid}

void setStartTime (int \textit{startTime})

Precondition:
\texttt{true}

Postcondition:
\texttt{this.startTime = startTime}
CHAPTER 3. COMPONENT DESCRIPTIONS

void setStatus (BookingStatus status)

Precondition:
   true

Postcondition:
   this.status = status

void setType (BookingType type)

Precondition:
   experimentID > -1

Postcondition:
   this.type = type

void store ()

Precondition:
   true

Postcondition:
   The data of the booking is stored into the database.

Member Data Documentation

Direction direction  [private]
   Direction of the communication between the MCS and the satellite.

int endTime  [private]

ExperimentID experimentID  [private]
   -1 if the booking is an internal booking, otherwise the experiment for which the external booking was made.

GSSID gssID  [private]

BookingID identifier  [private]

int startTime  [private]

BookingStatus status  [private]


3.2.18 Booking::BookingException Class Reference

#include <booking.h>
Inherits runtime_error.
Inheritance diagram for Booking::BookingException:

```
runtime_error
   |
   +-- Booking::BookingException

Public Member Functions

- BookingException (const std::string &arg)

Constructor & Destructor Documentation

BookingException (const std::string & arg) [inline]

3.2.19 BookingInfo Class Reference

Specifies a booking.
#include <defs.h>

Public Member Functions

- bool operator==(const BookingInfo &other) const
  Needed for testing.
Public Attributes

- std::string gssid
  
  Ground station id.

- Interval period
  
  Interval (p. 244) of the booking.

- std::string remarks
- bool twoway
  
  Specifies if it is twoway or not.

Member Function Documentation

bool operator==(const BookingInfo & other) const  [inline]

Member Data Documentation

std::string gssid

Interval period

std::string remarks

Remarks about the booking

bool twoway

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

- common/defs.h

3.2.20 cancelled Class Reference

Inherits xmlrpc_c::method.

Inheritance diagram for cancelled:
CHAPTER 3. COMPONENT DESCRIPTIONS

Public Member Functions

- cancelled (Satellite *tmpSat, MccController *tmpController)
- void execute (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)

Private Attributes

- MccController * controller
- Satellite * sat

Constructor & Destructor Documentation

cancelled (Satellite * tmpSat, MccController * tmpController) [inline]

Member Function Documentation

void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value * const retvalP) [inline]

Member Data Documentation

MccController* controller [private]

Satellite* sat [private]

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

- server/mccListener.cpp

3.2.21 changePassword Class Reference

Changes the password of an account.

#include <clientconnectivity.h>

Inherits xmlrpc_c::method.

Inheritance diagram for changePassword:

[Diagram of class hierarchy]

3D 0.1 159
Public Member Functions

- `changePassword` (Controller ∗tmpController)
- void `execute` (xmlrpc_c::paramList const &paramList, xmlrpc_c::value ∗const retvalP)

Private Attributes

- Controller ∗ controller

Detailed Description

Parameters:

- `session` The first parameter, the session that the change is requested for.
- `checksum` The checksum for the new password, being sha256(sha256(old password) + sha256(password))
- `passw` The new password, being sha256(password).

Constructor & Destructor Documentation

`changePassword` (Controller ∗ `tmpController`)

Member Function Documentation

void `execute` (xmlrpc_c::paramList const & `paramList`, xmlrpc_c::value ∗const `retvalP`)

Member Data Documentation

Controller ∗ controller [private]

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

- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.22 ClientConnectivity Class Reference

#include <clientconnectivity.h>

Inherits XmlRpcListener.
Inheritance diagram for ClientConnectivity:

Public Member Functions

- **ClientConnectivity** (int socket, Controller *controller)
  Constructor of ClientConnectivity (p. 160) object.

- **void run ()**
  Start listening for incoming messages.

- **~ClientConnectivity ()**
  Destructor of Client Connectivity object.

Protected Attributes

- xmlrpc_c::serverAbyss * `myAbyssServerPtr`
  The XML RPC server.

Private Attributes

- Controller * controller

Detailed Description

This class receives messages from the client and passes them on to the server.

Constructor & Destructor Documentation

**ClientConnectivity (int socket, Controller * controller)**

Parameters:

- **socket** The socket that will be used for listening.
- **controller** The controller that has to be called when receiving a message.

Precondition:

true
Member Function Documentation

void run () [inherited]

Precondition:
true

Postcondition:
The listeners is listening for incoming messages.

Member Data Documentation

Controller* controller [private]
Controller (p. 170) that created this connectivity and to which function calls are delegated.

xmlrpc_c::serverAbyss* myAbyssServerPtr [protected, inherited]
The documentation for this class was generated from the following files:
- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.23 Command Class Reference

< std::exception
#include <command.h>

Public Member Functions

- Command (const CommandID id) throw (CommandException)
  Constructor of a new command object with CommandID id.

- Command (const std::string &name, const PayloadID &payload, const CommandParam &commandParam)
  Constructor of a new command object, which is not yet in the database.

- Command ()
  Empty constructor.

- CommandID getID ()
  Returns the identifier for this command.

- std::string getName () throw (CommandException)
  Returns the name of the command.
CHAPTER 3. COMPONENT DESCRIPTIONS

- **CommandParam getParameters () throw (CommandException)**  
  Retrieves the command parameters.

- **PayloadID getPayloadID () throw (CommandException)**  
  Returns the payload to which the command belongs to.

- **void instantiate () throw (CommandException)**  
  Retrieves data from command with CommandID from lower layer.

- **void setName (const std::string &name)**  
  Sets the name of the command.

- **void setParameters (const CommandParam &commandParam)**  
  Sets the command parameters.

- **void setPayloadID (const PayloadID &payload)**  
  Sets the payload to which the command belongs to.

- **void setUpdate ()**  
  Indicate that the data needs to be updated.

- **void store () throw (CommandException)**  
  Stores internal data to the lower layer.

- **~Command ()**  
  Destructor of command object.

Private Attributes

- **CommandParam commandParam**  
  Parameters of the command.

- **CommandID identifier**  
  Identifier for this object in the lower layer.

- **std::string name**  
  Name of the command.

- **bool needs_storage**
- **bool needs_update**

- **PayloadID payload**  
  Payload (p. 276) to which the command belongs to.

Classes

- **class CommandException**  
  Exception class thrown by **Command** (p. 162).
CHAPTER 3. COMPONENT DESCRIPTIONS

Detailed Description
< std::string

Constructor & Destructor Documentation

Command ()

Precondition:
true

Postcondition:
this.identifier = -1 \ this.name = "" \ this.payload = -1 \ this.paramList = null \ this.needs_update = false \ this.needs_storage = false

Command (const std::string & name, const PayloadID & payload, const CommandParam & commandParam)

Parameters:
name a string representing the name of the command.

payload a PayloadID representing the payload to it belongs.

commandParam a CommandParam (p. 168) representing the parameters of the command.

Precondition:
true

Postcondition:
this.getPayload() = payload \ this.getName() = name \ this.getParameters() = commandParam \ this.needs_update = false \ this.needs_storage = false

Command (const CommandID id) throw (CommandException)

Exceptions:
CommandException (p. 168)

Parameters:

id a CommandID identifying the command in the lower layer.

Precondition:
id >= 0 \ (exists c in conn.command ( c.id = id))

Postcondition:
this.getID() = id \ this.name = "" \ this.payload = -1 \ this.needs_update = true \ this.needs_storage = false
CHAPTER 3. COMPONENT DESCRIPTIONS

~Command ()
Precondition:
   true

Member Function Documentation

CommandID getID ()
Precondition:
   true
Returns:
   identifier

string getName () throw (CommandException)
Exceptions:
   CommandException (p. 168)
Precondition:
   true
Returns:
   this.name

CommandParam getParameters () throw (CommandException)
Exceptions:
   CommandException (p. 168)
Precondition:
   true
Returns:
   this.commandParam

PayloadID getPayloadID () throw (CommandException)
Exceptions:
   CommandException (p. 168)
CHAPTER 3. COMPONENT DESCRIPTIONS

Precondition:
true
true

Returns:
this.payload

void instantiate () throw (CommandException)

Precondition:
identifier >= 0

Postcondition:
The parameters of the command are instantiated with information from the lower layer.

void setName (const std::string & name)

Parameters:
name a string representing the name of the command.

Precondition:
true

Postcondition:
this.getName() = name

void setParameters (const CommandParam & commandParam)

Parameters:
commandParam a CommandParam (p.168) consisting of the parameters of the command.

Precondition:
true

Postcondition:
this.commandParam = commandParam

void setPayloadID (const PayloadID & payload)

Parameters:
payload a payloadID representing the payload the command belongs to.
CHAPTER 3. COMPONENT DESCRIPTIONS

Precondition:
true

Postcondition:
this.getPayload() = payload

void setUpdate ()

Precondition:
this.identifier > -1

Postcondition:
this.needs_update = true

void store () throw (CommandException)

Exceptions:
CommandException (p. 168)

Precondition:
true
identifier >= 0

Postcondition:
The command entity with id is updated with the information from this class.

Member Data Documentation

CommandParam commandParam [private]

CommandID identifier [private]

std::string name [private]

bool needs_storage [private]
Whether the data of this payload needs client-server communication to store its attributes

bool needs_update [private]
Whether the data of this payload needs client-server communication to update its attributes

PayloadID payload [private]
The documentation for this class was generated from the following files:

- common/command.h
- common/command.cpp
3.2.24 Command::CommandException Class Reference

Exception class thrown by Command (p. 162).

```
#include <command.h>
```

Inherits `runtime_error`.

Inheritance diagram for Command::CommandException:

```
+ Command::CommandException
 |   + CommandException()
+ runtime_error
```

Public Member Functions

- `CommandException` (const std::string &arg)

Constructor & Destructor Documentation

`CommandException (const std::string & ary) [inline]`

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

- `common/command.h`

3.2.25 CommandParam Class Reference

Parameters of a command which can be used in an experiment.

```
#include <defs.h>
```

Public Member Functions

- `bool operator==(const CommandParam &other) const`

Public Attributes

- `ParamTypes defaultValue`
  
  Default value of the parameter.

- `ParamTypes maxValue`
  
  Maximum value of the parameter.

- `ParamTypes minValue`
  
  Minimum value of the parameter.
CHAPTER 3. COMPONENT DESCRIPTIONS

- **ParamType paramType**
  
  *Type of the parameter.*

Member Function Documentation

bool operator==(const CommandParam & other) const [inline]

< Needed for testing

Member Data Documentation

ParamTypes defaultValue

ParamTypes maxValue

ParamTypes minValue

**ParamType paramType**

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

- common/defs.h

### 3.2.26 ConnectionManager Class Reference

Provide a connection to the lower layer in order to abstract from the actual connection.

#include <connectionmanager.h>

**Static Public Member Functions**

- static AdminLowlevel * getConnection ()
  
  *Returns the connection this class manages.*

**Private Member Functions**

- ConnectionManager ()
  
  *Private constructor, necessary for Singleton.*

**Static Private Attributes**

- static AdminLowlevel * connection = NULL
CHAPTER 3. COMPONENT DESCRIPTIONS

Constructor & Destructor Documentation

ConnectionManager () [inline, private]

Member Function Documentation

AdminLowlevel * getConnection () [static]

Precondition:

    conn = this.connection

Postcondition:

    if conn = NULL -> this.connection = new AdminLowlevel() [] conn /= NULL -> skip fi

Returns:

    this.connection

Member Data Documentation

AdminLowlevel * connection = NULL [static, private]

Connection with the lower level of the system.

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

- common/connectionmanager.h
- common/connectionmanager.cpp

3.2.27 Controller Class Reference

#include <controller.h>

Public Types

- typedef std::pair<bool, AccountType> LoginInfo
  Type used in the return value of login (p. 247).

- typedef std::pair<SatelliteID, std::string> SatelliteInfo
  Return type of getSatelliteIDs.

Public Member Functions

- void addAccountToPayload (const std::string &session, const PayloadID &pid, const AccountID &aid) throw (ControllerException)
  Grant rights to payload pid to user aid.

- void addCommand (const std::string &session, const std::string &name, const PayloadID &payload, const CommandParam &commandParam) throw (ControllerException)
Add command for payload.

- void callbackExperimentUploadFailure (ExperimentID id)
  
  Callback function used to inform that an experiment will not be uploaded.

- void callbackListOfOpportunities (const SatelliteID id, const bool Status, const BookingList &listOfBookingInfos, const int requestid) throw (ControllerException)
  
  Callback function used for replies to getListOfopportunities.

- void callbackReceivedData (DataID id) throw (ControllerException)
  
  Callback function used to inform that new data has been received.

- bool changePassword (const std::string &session, const std::string &checksum, const std::string &password) throw (ControllerException)
  
  Changes the hashed version of the password belonging to session to password.

- Controller ()
  
  Default constructor.

- bool createAccount (const std::string &session, const AccountID &userName, const AccountType&type, const std::string &password, const std::string &fullName, const std::string &eMail) throw (ControllerException)
  
  Creates a new account with name userName if there is no such account. If such an account exists no account is added.

- ExperimentID createExperiment (const std::string &session, const std::string &name, const PayloadID &payload) throw (ControllerException)
  
  Create an Experiment (p. 221) on the server.

- PayloadID createPayload (const std::string &session, const std::string &name, const SatelliteID &satellite) throw (ControllerException)
  
  Creates a new payload.

- SatelliteID createSatellite (const std::string &session, const std::string &mccURL, const std::string &mcsURL, const std::string &name) throw (ControllerException)
  
  Creates a new satellite.

- void deleteExperiment (const std::string &session, const ExperimentID &id) throw (ControllerException)
  
  Deletes experiment e.

- std::pair< std::string, std::string > getChallenge ()
  
  Get a challenge and a session id.

- Command getCommand (const std::string &session, const CommandID &id) throw (ControllerException)
  
  Use this function to get the information of a command.

- Data getDatum (const std::string &session, const DataID &id) throw (ControllerException)
  
  Returns the data item with id id.
• **Experiment getExperiment** (const std::string &session, const **ExperimentID** &id) throw (ControllerException)
  
  *Use this function to get the experiment by ID.*

• std::vector< **ExperimentInfo** > getExperimentIDs (const std::string &session) throw (ControllerException)
  
  *Use this function to get the experiment IDs belong to this session.*

• **DataID getLastReceivedThumbnail** (const **ExperimentID** &id)
  
  *Get the id of the last thumbnail for experiment id that was received from the scheduler.*

• **DataID getLastRetrievedThumbnail** (const std::string &session, const **ExperimentID** &id)
  
  *Get the id of the last thumbnail for experiment id that was retrieved by session.*

• std::vector< **BookingInfo** > getListOfOpportunities (const std::string &session, const **ExperimentID** &id, const **Interval** &period)
  
  *Gets a list of opportunities on which the experiment with id can be approached for communication.*

• void getListOfThumbnails (const std::string &session, const **ExperimentID** &id, std::vector< **Data** > *result) throw (ControllerException)
  
  *Returns the list of thumbnails belonging to experiment id.*

• std::vector< std::string > getLog (const std::string &session, const **SatelliteID** &sid, const **PayloadID** &pid, const **ExperimentID** &eid, const int from, const int to) throw (ControllerException)
  
  *Returns the list of log entries for id, starting at time from, until time to.*

• **Payload getPayload** (const std::string &session, const **PayloadID** &id) throw (ControllerException)
  
  *Use this function to get the payload by ID.*

• std::vector< **PayloadInfo** > getPayloadIDs (const std::string &session) throw (ControllerException)
  
  *Use this function to get the payload IDs belonging to session.*

• **Satellite getSatellite** (const std::string &session, const **SatelliteID** &id) throw (ControllerException)
  
  *Use this function to get the satellite by ID.*

• std::vector< **SatelliteInfo** > getSatelliteIDs (const std::string &session) throw (ControllerException)
  
  *Use this function to get the satellite IDs belonging to the session.*

• bool hasHighResolution (const std::string &session, const **DataID** &id) throw (ControllerException)
  
  *Returns whether the high resolution version of the datum with identifier id is available on the server.*
CHAPTER 3. COMPONENT DESCRIPTIONS

- **LoginInfo login** (const std::string &session, const std::string &name, const std::string &password) throw (ControllerException)
  
  Use this function to log in the user with name and password.

- **bool logout** (const std::string &session, const std::string &name) throw (ControllerException)
  
  Use this function to log out the user with name and password.

- **void markForDeletion** (const std::string &session, const DataID &id) throw (ControllerException)
  
  Marks the data item with id to be deleted from the satellite.

- **void markForHighResolutionDownload** (const std::string &session, const DataID &id) throw (ControllerException)
  
  Marks the high resolution version of the datum with identifier id for download.

- **void registerAccount** (const std::string &session, const AccountID &userName) throw (ControllerException)

- **void registerPayload** (const std::string &session, const PayloadID &id) throw (ControllerException)
  
  Registers the payload with the system.

- **void registerSatellite** (const std::string &session, const SatelliteID &id) throw (ControllerException)
  
  Registers the satellite with the system.

- **void removeAccount** (const std::string &session, const AccountID &userName) throw (ControllerException)
  
  Removes the account with name userName.

- **void removeAccountFromPayload** (const std::string &session, const PayloadID &pid, const AccountID &aid) throw (ControllerException)
  
  Revoke rights to payload pid for user aid.

- **void removeCommand** (const std::string &session, const std::string &name, const PayloadID &payload) throw (ControllerException)
  
  Remove command from payload.

- **void removePayload** (const std::string &session, const PayloadID &id) throw (ControllerException)
  
  Remove the payload from the system.

- **void removeSatellite** (const std::string &session, const SatelliteID &id) throw (ControllerException)
  
  Removes the satellite with id from the system.

- **Data requestThumbnail** (const std::string &session, const ExperimentID &id) throw (ControllerException)
  
  Returns the next thumbnail for experimentID.
CHAPTER 3. COMPONENT DESCRIPTIONS

- **bool scheduleExperiment** (const std::string &session, const ExperimentID &id, const int startTime, const std::vector<BookingInfo> &booking) throw (ControllerException)

  Schedule the Experiment (p. 221) for execution with the scheduler. This means that if the experiment has not yet been uploaded, the experiment to be uploaded is either added or updated.

- **void sendCommand** (const std::string &session, const ExperimentID &id, const Command &c) throw (ControllerException)

  Queues (p. 290) command c to be executed for the payload of experiment id.

- **void unregisterAccount** (const std::string &session, const AccountID &userName) throw (ControllerException)

- **void unregisterPayload** (const std::string &session, const PayloadID &id) throw (ControllerException)

  Unregisters the payload with the system.

- **void unregisterSatellite** (const std::string &session, const SatelliteID &id) throw (ControllerException)

  Unregisters the satellite with the system.

- **void unscheduleExperiment** (const std::string &session, const ExperimentID &id) throw (ControllerException)

  Try to unschedule the experiment with ID id, note that there may be a race condition, as inbetween the time the unschedule command reaches the satellite, the experiment might have been executed.

- **void updateAccount** (const std::string &session, const AccountID &userName, const AccountType &type, const std::string &password, const std::string &fullName, const std::string &eMail) throw (ControllerException)

- **void updateCommand** (const std::string &session, const std::string &name, const PayloadID &payload, const CommandParam &commandParam) throw (ControllerException)

  Update command for payload.

- **bool updateExperiment** (const std::string &session, const ExperimentID &id, const std::string &name, const int scheduledTo, ListOfExperimentCommands expCommands) throw (ControllerException)

  Updates the Experiment (p. 221) on the server.

- **void updatePayload** (const std::string &session, const PayloadID &id, const std::string &name, const SatelliteID &satellite) throw (ControllerException)

  Updates the payload with id.

- **void updateSatellite** (const std::string &session, const SatelliteID &id, const std::string &mccURL, const std::string &mcsURL, const std::string &name) throw (ControllerException)

  Creates a new satellite.

- **~Controller ()**

  Default destructor.
CHAPTER 3. COMPONENT DESCRIPTIONS

Public Attributes

- std::map<int, CallbackHandleOpportunities> callbackListOpportunities
  - List of requested callbacks.

- ThreadingInformation waitCallbackData
  - Condition variable used to wait for callback with data.

- ThreadingInformation waitCallbackOpportunities
  - Condition variable used to wait for callback with list of.

Protected Member Functions

- void cleanExpiredSessions()
  - Cleans up sessions that have been timed out.

- Account getAccountFromSession(const std::string &session) const
  - Get the account that is attached to session.

- bool isValidSession(const std::string &session)
  - Check that there is a session session.

- void updateKeepalive(const std::string &session)
  - Updates the lastKeepalive of session to now.

Protected Attributes

- std::map<ExperimentID, DataID> receivedThumbnails
  - Thumbnail administration.

- Scheduler scheduler
  - Scheduler (p. 302).

- std::map<std::string, SessionInfo> sessions
  - Session administration.

Classes

- class CallbackHandleOpportunities
  - Callback handle for requesting list of opportunities.

- class ControllerException
  - Exception thrown by the Controller (p. 170) class.

- class ExperimentInfo
  - Class used in the return value of getExperimentIDs.
• class **PayloadInfo**  
  Class used in the return value of `getPayloadIDs`.

• class **SessionInfo**  
  Simple structure that is used to keep the information belonging to a session.

• class **ThreadArgumentInt**  
  Structure used to pass controller and id into a thread.

• class **ThreadArgumentStringInt**  
  Structure used to pass controller, id and session into a thread.

• class **ThreadingInformation**  
  Administration for threads.

**Detailed Description**

Class that distributes incoming requests from the client over the various components of the server.

**Member Typedef Documentation**

typedef `std::pair<bool, AccountType>` **LoginInfo**

typedef `std::pair<SatelliteID, std::string>` **SatelliteInfo**

**Constructor & Destructor Documentation**

Controller ()

~Controller ()

**Member Function Documentation**

void **addAccountToPayload** (const `std::string & session`, const `PayloadID & pid`, const `AccountID & aid`) throw (`ControllerException`)

**Exceptions:**

  *ControllerException* (p. 194)

**Parameters:**

  - `session` The session that grants the rights.
  - `pid` The identifier of the payload to which rights are granted.
  - `aid` The identifier of the account who rights are granted to.

**Todo**

Implement using **Payload** (p. 276) class.
void addCommand (const std::string & session, const std::string & name, const PayloadID & payload, const CommandParam & commandParam) throw (ControllerException)

Exceptions:

ControllerException (p. 194)

Parameters:

session The session that adds the command.

name The name of the command.

payload The payload on which the command is defined.

commandParam The parameter of the command.

Todo

Implement using Command (p. 162) class.

void callbackExperimentUploadFailure (ExperimentID id)

Parameters:

id Id of the experiment that will not be uploaded.

Todo

This is currently unused, but could facilitate a negative answer to the client.

void callbackListOfOpportunities (const SatelliteID id, const bool Status, const BookingList & listOfBookingInfos, const int requestid) throw (ControllerException)

Exceptions:

ControllerException (p. 194)

Parameters:

id The satelliteid to which this callback applies.

Status whether the request was succesful.

listOfBookingInfos A list of available passes.

requestid the id to which this callback is a reply to.

void callbackReceivedData (DataID id) throw (ControllerException)

Exceptions:

ControllerException (p. 194)

Parameters:

id Id of the data received.
bool changePassword (const std::string & session, const std::string & checksum, const std::string & password) throw (ControllerException)

Parameters:

- **session** The session that the password change is requested for.
- **checksum** The checksum for the new password, being sha256(sha256(old password) + sha256(new password)).
- **password** The hashed new password, being sha256(new password).

void cleanExpiredSessions () [protected]

bool createAccount (const std::string & session, const AccountID & userName, const AccountType & type, const std::string & password, const std::string & fullName, const std::string & eMail) throw (ControllerException)

Exceptions:

- **ControllerException** (p. 194)

Parameters:

- **session** The session that creates the account.
- **userName** The userName for the new account.
- **type** The type of the new account.
- **password** The password of the new account.
- **fullName** The full name belonging to the account.
- **eMail** The e-mail address belonging to the account.

Returns:

- true if the new account was successfully added, false otherwise.

Todo

Implement using **Account** (p. 114) class.

ExperimentID createExperiment (const std::string & session, const std::string & name, const PayloadID & payload) throw (ControllerException)

Exceptions:

- **ControllerException** (p. 194)

Parameters:

- **session** The session requesting the create.
- **name** The name of the experiment to be created.
- **payload** The identifier of the payload for which the experiment is created.
CHAPTER 3. COMPONENT DESCRIPTIONS

Precondition:

\[ \text{payload} \implies 0 \lor \neg \exists e \in \text{Experiment} \ (\text{p. 221}): e.\text{payload} = \text{payload}: e.\text{name} = \text{name}. \]

Postcondition:

\text{Experiment} \ (\text{p. 221}) \ is \ created \ on \ \text{payload} \ with \ \text{name} \ and \ a \ fresh \ identifier, \ \text{id}.

Returns:

\text{id}

PayloadID createPayload (const std::string & \text{session}, const std::string & \text{name}, const SatelliteID & \text{satellite}) throw (ControllerException)

Exceptions:

\text{ControllerException} \ (\text{p. 194})

Parameters:

\text{session} \ The \ session \ that \ creates \ the \ payload.

\text{name} \ The \ name \ of \ the \ payload.

\text{satellite} \ The \ satellite \ the \ payload \ belongs \ to.

Returns:

The \ id \ of \ the \ payload.

Todo

Implement \ using \ \text{Payload} \ (\text{p. 276}) \ class.

SatelliteID createSatellite (const std::string & \text{session}, const std::string & \text{mccURL}, const std::string & \text{mcsURL}, const std::string & \text{name}) throw (ControllerException)

Exceptions:

\text{ControllerException} \ (\text{p. 194})

Parameters:

\text{session} \ The \ session \ that \ creates \ the \ satellite.

\text{mccURL} \ The \ URL \ of \ the \ MCC \ for \ the \ satellite.

\text{mcsURL} \ The \ URL \ of \ the \ MCS \ for \ the \ satellite.

\text{name} \ The \ name \ of \ the \ satellite.

Returns:

The \ id \ of \ the \ satellite.

Todo

Implement \ using \ \text{Satellite} \ (\text{p. 293}) \ class.

DDD 0.1 179
void deleteExperiment (const std::string & session, const ExperimentID & id) throw (ControllerException)

Exceptions:

ControllerException (p. 194)

Parameters:

session The session that requests deletion of the experiment.

id The id of the experiment to be deleted.

Precondition:

The experiment has not yet been uploaded.

Postcondition:

The experiment has been deleted.

Todo

Implement deletion of experiment using the Experiment (p. 221) class.

Account getAccountFromSession (const std::string & session) const  [protected]

Parameters:

session The session the request belongs to.

Precondition:

session is a valid session.

Returns:

Account (p. 114)(sessions[session].accountID).

std::pair< std::string, std::string > getChallenge ()

Precondition:

true

Returns:

(challenge, sessionid).

Postcondition:

A session with sessionid is added to sessions.

I guess a higher number doesn’t make it stronger,

Todo

we need real random data.
CHAPTER 3. COMPONENT DESCRIPTIONS

Command getCommand (const std::string & session, const CommandID & id) throw (ControllerException)

Parameters:

  * session The session requesting the command.
  * id The identifier of the command.

Precondition:

  commandID >= 0

Returns:

  the Command (p. 162) c such that c.getID() = id.

Data getDatum (const std::string & session, const DataID & id) throw (ControllerException)

Exceptions:

  ControllerException (p. 194)

Parameters:

  * session The session that requests the data.
  * id The id of the data that is requested.

Returns:

  The data item with identifier id.

Todo

  Implement by checking the user has rights to the experiment the data belongs to. Then get the data and instantiate it. Throw an exception if no such data exists.

Experiment getExperiment (const std::string & session, const ExperimentID & id) throw (ControllerException)

Parameters:

  * session The session that requests the experiment.
  * id The identifier of the experiment.

Precondition:

  id >= 0

Returns:

  Experiment (p. 221) e such that e.getID() = id.
vector<Controller::ExperimentInfo> getExperimentIDs (const std::string & session) throw (ControllerException)

Parameters:

*session* The session that requests the experimentIDs.

Exceptions:

*ControllerException (p. 194)*

Precondition:

Session belongs to a user (or true?).

Returns:

The experimentIDs of experiments to the user with session.

DataID getLastReceivedThumbnail (const ExperimentID & id)

DataID getLastRetrievedThumbnail (const std::string & session, const ExperimentID & id)

std::vector<BookingInfo> getListOfOpportunities (const std::string & session, const ExperimentID & id, const Interval & period)

Parameters:

*session* The session requesting the list of opportunities.

*id* The identifier of the experiment.

*period* The period in which opportunities are requested.

Precondition:

id >= 0

Returns:

[b: BookingInfo (p. 157) | exists e: Experiment (p. 221) : e.identifier = id /\ e.Payload.Satellite passes during b].

void getListOfThumbnails (const std::string & session, const ExperimentID & id, std::vector<Data> * result) throw (ControllerException)

Parameters:

*session* The session that requests the list of thumbnails.

*id* The id for which the list of thumbnails is requested.

*result* The list of thumbnails.

Returns:

The list of thumbnails belonging to id.
std::vector< std::string > getLog (const std::string & session, const SatelliteID & sid, const PayloadID & pid, const ExperimentID & eid, const int from, const int to) throw (ControllerException)

Parameters:

- **session**: The session that requests the log.
- **sid**: The id of the satellite for which we want to retrieve the log.
- **pid**: The id of the payload for which we want to retrieve the log.
- **eid**: The id of the experiment for which we want to retrieve the log.
- **from**: The start time from which the logs are desired.
- **to**: The end time from which the logs are desired.

Returns:

The logs entries for satellite id, for the period [from..to].

Todo

Implement retrieval of logs using the Log class.

Payload getPayload (const std::string & session, const PayloadID & id) throw (ControllerException)

Parameters:

- **session**: The session that requests the payload.
- **id**: The identifier of the payload.

Exceptions:

*ControllerException* (p. 194)

Precondition:

id $\geq 0$

Returns:

Payload (p. 276) p such that p.getID() = id.

vector< Controller::PayloadInfo > getPayloadIDs (const std::string & session) throw (ControllerException)

Parameters:

- **session**: The session that requests the payloadIDs.

Exceptions:

*ControllerException* (p. 194)
Precondition:

Session belongs to a user (or true?).

Returns:

The payloads belonging to the session.

Satellite getSatellite (const std::string & session, const SatelliteID & id) throw (ControllerException)

Parameters:

session The session that requests the satellite.

id The identifier of the satellite.

Exceptions:

ControllerException (p. 194)

Precondition:

id >= 0

Returns:

Satellite (p. 293) s such that s.getID() = id.

vector<Controller::SatelliteInfo> getSatelliteIDs (const std::string & session) throw (ControllerException)

Parameters:

session The session for which we want to retrieve the satellite IDs.

Exceptions:

ControllerException (p. 194)

Precondition:

ture

Postcondition:

All SatelliteIDs of satellites for which there is an experiment to which the user belonging to session has sufficient rights.
 CHAPTER 3. COMPONENT DESCRIPTIONS

Parameters:

- `session` The session that does the request.
- `id` The id of the data the check is done for.

Returns:

true if the high-resolution version of the datum with id is available on the server, false otherwise.

Todo

Implement using Data (p. 199) class.

```cpp
bool isValidSession (const std::string & session) [protected]
```

Parameters:

- `session` The session for which we want to check if it is valid.

Precondition:

true

Returns:

true if there is a session with id session, false otherwise.

```cpp
Controller::LoginInfo login (const std::string & session, const std::string & name, const std::string & password) throw (ControllerException)
```

Parameters:

- `session` The session that the login (p. 247) is requested for.
- `name` The user name of the user that wants to login (p. 247).
- `password` The password of the user.

Exceptions:

- `ControllerException` (p. 194)

Precondition:

true

Postcondition:

If the user name and password match, the user is logged in, and the user name is linked to the session.

Returns:

true if the login (p. 247) succeeded, false otherwise.
bool logout (const std::string & session, const std::string & name) throw (ControllerException)

Parameters:

    session The session that the logout (p. 249) is requested for.
    name The user name of the user that wants to log out.

Exceptions:

    ControllerException (p. 194)

Precondition:

ture

Postcondition:

    If there is a session session to which the user with user name name is linked the session is removed. Otherwise there is no change.

Returns:

    true if there is no session with id session or the user with user name name is linked to the session, false otherwise.

void markForDeletion (const std::string & session, const DataID & id) throw (ControllerException)

Exceptions:

    ControllerException (p. 194)

Parameters:

    session The session that requests the marking.
    id The id of the data that needs to be deleted.

Todo

    Implement marking for deletion of data using the Data (p. 199) class.

void markForHighResolutionDownload (const std::string & session, const DataID & id) throw (ControllerException)

Exceptions:

    ControllerException (p. 194)

Parameters:

    session The session that does the marking.
    id The id of the datum that is marked.

Todo

    Implement using Data (p. 199) class.
void registerAccount (const std::string & session, const AccountID & userName) throw (ControllerException)  

Todo  
Implement using Account (p. 114) class.

void registerPayload (const std::string & session, const PayloadID & id) throw (ControllerException)  

Exceptions:  
ControllerException (p. 194)  

Parameters:  
   session The session that registers the payload.  
   id The id of the payload that is registered.  

Todo  
Implement using Payload (p. 276) class.

void registerSatellite (const std::string & session, const SatelliteID & id) throw (ControllerException)  

Exceptions:  
ControllerException (p. 194)  

Parameters:  
   session The session that registers the satellite.  
   id The id of the satellite that is registered.  

Todo  
Implement using Satellite (p. 293) class.

void removeAccount (const std::string & session, const AccountID & userName) throw (ControllerException)  

Exceptions:  
ControllerException'

Parameters:  
   session The session that deletes the account.  
   userName The userName for the account that is deleted.  

Todo  
Implement using Account (p. 114) class.
void removeAccountFromPayload (const std::string & session, const PayloadID & pid, const AccountID & aid) throw (ControllerException)

Exceptions:

ControllerException (p. 194)

Parameters:

session The session that revokes the rights.

pid The identifier of the payload to which rights are revoked.

aid The identifier of the account who the rights are revoked from.

Todo

Implement using Payload (p. 276) class.

void removeCommand (const std::string & session, const std::string & name, const PayloadID & payload) throw (ControllerException)

Exceptions:

ControllerException (p. 194)

Parameters:

session The session that removes the command.

name The name of the command.

payload The payload on which the command is defined.

Todo

Implement using Command (p. 162) class.

void removePayload (const std::string & session, const PayloadID & id) throw (ControllerException)

Exceptions:

ControllerException (p. 194)

Parameters:

session The session that removes the payload.

id The id of the payload that is removed.

Todo

Implement using Payload (p. 276) class.
void removeSatellite (const std::string & session, const SatelliteID & id) throw (ControllerException)

Exceptions:

ControllerException (p. 194)

Parameters:

session The session that removes the satellite.
id The id of the satellite that is deleted.

Todo

Implement using Satellite (p. 293) class.

Data requestThumbnail (const std::string & session, const ExperimentID & id) throw (ControllerException)

Exceptions:

ControllerException (p. 194)

Parameters:

session The session requesting the thumbnail.
id The identifier of the experiment for which the thumbnail is requested.

bool scheduleExperiment (const std::string & session, const ExperimentID & id, const int startTime, const std::vector< BookingInfo > & booking) throw (ControllerException)

Parameters:

session The session that requests the scheduling.
id The identifier of the experiment to be scheduled.
startTime The time the experiment is scheduled to start.
booking The list of opportunities that need to be booked (p. 149) for this experiment.

Returns:

true if scheduling succeeded, false otherwise.

void sendCommand (const std::string & session, const ExperimentID & id, const Command & c) throw (ControllerException)

Exceptions:

ControllerException (p. 194)
CHAPTER 3. COMPONENT DESCRIPTIONS

Parameters:

\textit{session} The session that requests the execution of the command.

\textit{id} The id of the experiment for which the command has to be executed.

\textit{c} The command that has to be executed.

Todo

Implement by checking for the appropriate rights (similar to uploadExperiment), and using the executeCommand function from the scheduler.

\begin{verbatim}
void unregisterAccount (const std::string & session, const AccountID & userName) throw (ControllerException)
\end{verbatim}

Todo

Implement using \texttt{Account} (p. 114) class.

\begin{verbatim}
void unregisterPayload (const std::string & session, const PayloadID & id) throw (ControllerException)
\end{verbatim}

Exceptions:

\textit{ControllerException} (p. 194)

Parameters:

\textit{session} The session that unregisters the payload.

\textit{id} The id of the payload that is unregistered.

Todo

Implement using \texttt{Payload} (p. 276) class.

\begin{verbatim}
void unregisterSatellite (const std::string & session, const SatelliteID & id) throw (ControllerException)
\end{verbatim}

Exceptions:

\textit{ControllerException} (p. 194)

Parameters:

\textit{session} The session that unregisters the satellite.

\textit{id} The id of the satellite that is unregistered.

Todo

Implement using \texttt{Satellite} (p. 293) class.
CHAPTER 3. COMPONENT DESCRIPTIONS

void unscheduleExperiment (const std::string & session, const ExperimentID & id) throw (ControllerException)

Exceptions:

  ControllerException (p. 194)

Parameters:

  session The session that requests the unscheduling.
  id The identifier of the experiment to be unscheduled.

Returns:

  true if unscheduling succeeded, false otherwise.

Todo

  Check when unscheduling may happen!

void updateAccount (const std::string & session, const AccountID & userName, const AccountType & type, const std::string & password, const std::string & fullName, const std::string & eMail) throw (ControllerException)

Todo

  Implement using Account (p. 114) class.

void updateCommand (const std::string & session, const std::string & name, const PayloadID & payload, const CommandParam & commandParam) throw (ControllerException)

Exceptions:

  ControllerException (p. 194)

Parameters:

  session The session that adds the command.
  name The name of the command.
  payload The payload on which the command is defined.
  commandParam The parameter of the command.

Todo

  Implement using Command (p. 162) class.
bool updateExperiment (const std::string & session, const ExperimentID & id, const std::string & name, const int scheduledTo, ListOfExperimentCommands expCommands) throw (ControllerException)

Parameters:

    session  The session that requests the update.
    id       The identifier of the experiment.
    name     The name of the experiment.
    scheduledTo  The time to which the experiment is scheduled to run.
    expCommands  The sequence of commands the experiment consists of.

Returns:

    true if the update was successful, false otherwise.

void updateKeepalive (const std::string & session)  [protected]

Parameters:

    session  The session for which the keepalive is updated.

Postcondition:

    The current time for session is updated to a time approx. now.

void updatePayload (const std::string & session, const PayloadID & id, const std::string & name, const SatelliteID & satellite) throw (ControllerException)

Exceptions:

    ControllerException (p. 194)

Parameters:

    session  The session that updates the payload.
    id       The id of the payload that is updated.
    name     The name of the payload.
    satellite  The satellite the payload belongs to.

Returns:

    The id of the payload.

Todo

    Implement using Payload (p. 276) class.
void updateSatellite (const std::string & session, const SatelliteID & id, const std::string & mccURL, const std::string & mcsURL, const std::string & name) throw (ControllerException)

Exceptions:

ControllerException (p. 194)

Parameters:

session The session that updates the satellite.
id The id of the satellite that is updated.
mccURL The URL of the MCC for the satellite.
mcsURL The URL of the MCS for the satellite.
name The name of the satellite.

Todo

Implement using Satellite (p. 293) class.

Member Data Documentation

std::map<int, CallbackHandleOpportunities> callbackListOpportunities

std::map<ExperimentID, DataID> receivedThumbnails [protected]

Scheduler scheduler [protected]

std::map<std::string, SessionInfo> sessions [protected]

ThreadingInformation waitCallbackData

ThreadingInformation waitCallbackOpportunities

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

- server/controller.h
- server/controller.cpp

3.2.28 Controller::CallbackHandleOpportunities Class Reference

Callback handle for requesting list of opportunities.

#include <controller.h>

Public Attributes

- BookingList bookings
- bool completed
- ExperimentID id
- bool status
CHAPTER 3. COMPONENT DESCRIPTIONS

Member Data Documentation

BookingList bookings

bool completed

ExperimentID id

bool status

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

- server/controller.h

3.2.29 Controller::ControllerException Class Reference

Exception thrown by the Controller (p. 170) class.

#include <controller.h>

Inherits runtime_error.

Inheritance diagram for Controller::ControllerException:

```
+ ControllerException()
 runtime_error
```

Public Member Functions

- ControllerException (const std::string &arg)
  
  Throw controller exception.

Constructor & Destructor Documentation

ControllerException (const std::string & arg) [inline]

Parameters:

arg The exception message.

Postcondition:

The error is logged and an exception is thrown.
CHAPTER 3. COMPONENT DESCRIPTIONS

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

- server/controller.h

### 3.2.30 Controller::ExperimentInfo Class Reference

Class used in the return value of getExperimentIDs.

```cpp
#include <controller.h>
```

**Public Attributes**

- **ExperimentID identifier**
  
  Identifier of the experiment.

- **std::string name**
  
  Name of the experiment.

- **PayloadID payload**
  
  Identifier of the payload the experiment with id ExperimentID belongs to.

- **ExperimentStatus status**
  
  Status of the experiment.

**Member Data Documentation**

**ExperimentID identifier**

```cpp
std::string name
```

**PayloadID payload**

**ExperimentStatus status**

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

- server/controller.h

### 3.2.31 Controller::PayloadInfo Class Reference

Class used in the return value of getPayloadIDs.

```cpp
#include <controller.h>
```
CHAPTER 3. COMPONENT DESCRIPTIONS

Public Attributes

- **PayloadID identifier**
  
  *Identifier of the payload.*

- **std::string name**
  
  *Name of the payload.*

- **SatelliteID satelliteID**
  
  *Identifier of the satellite the payload with id PayloadID belongs to.*

Member Data Documentation

**PayloadID identifier**

**std::string name**

**SatelliteID satelliteID**

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

- server/controller.h

### 3.2.32 Controller::SessionInfo Class Reference

Simple structure that is used to keep the information belonging to a session.

```cpp
#include <controller.h>
```

Public Attributes

- **AccountID accountID**

- **std::string challenge**

- **time_t lastKeepalive**

- **std::map< ExperimentID, DataID > lastRetrievedThumbnail**

Member Data Documentation

**AccountID accountID**

**std::string challenge**

**time_t lastKeepalive**

**std::map< ExperimentID, DataID > lastRetrievedThumbnail**

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

- server/controller.h
CHAPTER 3. COMPONENT DESCRIPTIONS

3.2.33 Controller::ThreadArgumentInt Class Reference

Structure used to pass controller and id into a thread.

#include <controller.h>

Public Attributes

- Controller * controller
- int * id

Member Data Documentation

Controller* controller

int* id

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

- server/controller.h

3.2.34 Controller::ThreadArgumentStringInt Class Reference

Structure used to pass controller, id and session into a thread.

#include <controller.h>

Public Attributes

- Controller * controller
- const int * id
- const std::string * session

Member Data Documentation

Controller* controller

const int* id

const std::string* session

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

- server/controller.h

3.2.35 Controller::ThreadingInformation Class Reference

Administration for threads.

#include <controller.h>

DDD 0.1 197
CHAPTER 3. COMPONENT DESCRIPTIONS

Public Attributes

- pthread_cond_t * condition
- pthread_mutex_t * mutex

Member Data Documentation

pthread_cond_t* condition

pthread_mutex_t* mutex

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

- server/controller.h

3.2.36 createExperiment Class Reference

Creates a new experiment for a payload with payloadID and with a name.

#include <clientconnectivity.h>

Inherits xmlrpc_c::method.

Inheritance diagram for createExperiment:

```
method

createExperiment

- controller
+ createExperiment()
+ execute()
```

Public Member Functions

- createExperiment (Controller *tmpController)
- void execute (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)

Private Attributes

- Controller * controller

Detailed Description

Parameters:

- **session** The first parameter, the session requesting the creation.
- **name** Second parameter, the name of the new experiment
- **payload** Third parameter, the PayloadID of the payload for which the experiment should be created.
Constructor & Destructor Documentation

createExperiment (Controller ∗ tmpController)

Member Function Documentation

void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value ∗ const retvalP)

Member Data Documentation

Controller∗ controller [private]

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

- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.37 Data Class Reference

< std::exception Administration frontend for managing data entities.

#include <data.h>

Public Member Functions

- Data (const Data &d)
- Data (const DataID id) throw (DataException)
  Constructor of a new data object with DataID id.
- Data (const std::string &name, const ExperimentID &experimentID)
  Constructor of a new data object, which is not yet in the database.
- Data ()
  Empty constructor.
- char ∗ getDataItem () throw (DataException)
  Returns the data item.
- int getDataItemLength () throw (DataException)
  Returns the length of the data item.
- ExperimentID getExperimentID () throw (DataException)
  Returns the identifier of the experiment to which the data is linked.
- int getGatheredTime () throw (DataException)
  Returns the time at which the data is gathered.
- DataID getID () const
  Returns the identifier for this data object.
CHAPTER 3. COMPONENT DESCRIPTIONS

- **bool getMarkedForDeletion () throw (DataException)**
  Returns if the data is marked for deletion or not.

- **std::string getName () throw (DataException)**
  Returns the name of the data object.

- **char * getThumbnail () throw (DataException)**
  Returns the thumbnail.

- **int getThumbnailLength () throw (DataException)**
  Returns the length of the data item.

- **std::string getType () throw (DataException)**
  Returns the type of the data.

- **void instantiate () throw (DataException)**
  Retrieves data from data with DataID from lower layer.

- **void operator= (const Data &d)**
- **void setDataItem (char *data, const int length)**
  Sets the dataitem.

- **void setDataItemLength (const int &length)**
  Sets the length of the data item.

- **void setExperimentID (const ExperimentID &experiment)**
  Links the data object to the experiment with identifier experiment.

- **void setGatheredTime (const int &time)**
  Sets the time at which the data is gathered.

- **void setMarkedForDeletion (const bool value)**
  Mark or unmark the data for deletion.

- **void setName (const std::string &name)**
  Sets the name of the data object.

- **void setThumbnail (char *data, const int length)**
  Sets the dataitem.

- **void setThumbnailLength (const int &length)**
  Sets the length of the thumbnail.

- **void setType (const std::string &type)**
  Sets the type of the data.

- **void setUpdate ()**
  Indicate that the data needs to be updated.
void store () throw (DataException)

Stores the data of the data object into the database.

~Data ()

Destructor of data object.

Private Attributes

- char * dataItem
  Highres version of data item.

- int dataItemLength
  Length of data item.

- ExperimentID experiment
  The ExperimentID this data item belongs to.

- int gatheredAt
  Time the data item is gathered at.

- DataID identifier
  Identifier for this object in the lower layer.

- bool markedForDeletion
  If the data is marked for deletion or not.

- std::string name
  Name of the data item.

- bool needs_storage
- bool needs_update
- char * thumbnail
  Thumbnail version of data item.

- int thumbnailLength
  Length of thumbnail item.

- std::string type
  Type of the data.

Classes

- class DataException
  Exception class thrown by Data (p. 199).
CHAPTER 3. COMPONENT DESCRIPTIONS

Constructor & Destructor Documentation

Data ()

Precondition:
true

Postcondition:
\[\text{this.getName()} = "" \lor \text{this.getExperimentID()} = -1 \lor \text{this.getGatheredTime()} = -1 \lor \text{this.getThumbnailLength()} = -1 \lor \text{this.getDataLength()} = -1 \lor \text{this.getType()} = "" \lor \text{this.needs_update} = false \lor \text{this.needs_storage} = false\]

Data (const std::string & name, const ExperimentID & experimentID)

Parameters:
\text{name} The name of the data (the path).
\text{experimentID} Representing the experiment the data belongs to.

Precondition:
true

Postcondition:
\[\text{this.getName()} = \text{name} \lor \text{this.getGatheredTime()} = -1 \lor \text{this.getExperimentID()} = \text{experimentID} \lor \text{this.getID()} = -1 \lor \text{this.getThumbnailLength()} = -1 \lor \text{this.getDataLength()} = -1 \lor \text{this.getType()} = "" \lor \text{this.needs_update} = false \lor \text{this.needs_storage} = false\]

Note: after this function instantiate has to be called! The tuple name, experimentID is unique.
When this tuple exists in the database, then instantiate it, else add it to the database.

Data (const DataID id) throw (DataException)

Exceptions:
\text{DataException} (p. 209)

Parameters:
id A DataID identifying the data in the lower layer.

Precondition:
id \geq 0 \lor (\exists \text{d in connection.Data (d.id = id)})

Postcondition:
\[\text{this.getName()} = "" \lor \text{this.getGatheredTime()} = -1 \lor \text{this.getExperimentID()} = -1 \lor \text{this.getID()} = -1 \lor \text{this.getThumbnailLength()} = -1 \lor \text{this.getDataLength()} = -1 \lor \text{this.getType()} = "" \lor \text{this.needs_update} = true \lor \text{this.needs_storage} = false\]
CHAPTER 3. COMPONENT DESCRIPTIONS

Data (const Data & d)

~Data ()

Todo

Clean up dataItem and thumbnail? What if this is copy-constructed?

Member Function Documentation

char * getDataItem () throw (DataException)

Exceptions:

    DataException (p. 209)

Precondition:

    true

Returns:

    this.dataItem

int getDataItemLength () throw (DataException)

Exceptions:

    DataException (p. 209)

Precondition:

    true

Returns:

    this.dataItemLength

ExperimentID getExperimentID () throw (DataException)

Exceptions:

    DataException (p. 209)

Precondition:

    true

Returns:

    this.experiment
CHAPTER 3. COMPONENT DESCRIPTIONS

int getGatheredTime () throw (DataException)

Exceptions:
   DataException (p. 209)

Precondition:
   true

Returns:
   this.gatheredAt

DataIDgetID () const

Precondition:
   true

Returns:
   this.identifier

bool getMarkedForDeletion () throw (DataException)

Exceptions:
   DataException (p. 209)

Precondition:
   true

Returns:
   this.markedForDeletion

string getName () throw (DataException)

Exceptions:
   DataException (p. 209)

Precondition:
   true

Returns:
   this.name
char * getThumbnail () throw (DataException)

Exceptions:

\[ \text{DataException (p. 209)} \]

Precondition:

true

Returns:

this.thumbnail

int getThumbnailLength () throw (DataException)

Exceptions:

\[ \text{DataException (p. 209)} \]

Precondition:

true

Returns:

this.dataItemLength

string getType () throw (DataException)

Exceptions:

\[ \text{DataException (p. 209)} \]

Precondition:

true

Returns:

this.data

void instantiate () throw (DataException)

Exceptions:

\( (\text{DataException}) \)

Precondition:

\( \text{this.getID()} \geq 0 \)

Postcondition:

The parameters of the data are instantiated with information from the lower layer /\ this.needs_ update = false
void operator= (const Data & d)

void setDataItem (char * data, const int length)

Parameters:

data The data item.

length An integer representing the length of the thumbnail.

Precondition:

true

Postcondition:

this.getThumbnailLength() = length \ this.getDataItem() = data \ this.needs_storage = true

void setDataItemLength (const int & length)

Parameters:

length An integer representing the length of the data item.

Precondition:

true

Postcondition:

this.getDataItemLength() = length \ this.needs_storage = true

void setExperimentID (const ExperimentID & experiment)

Precondition:

ture

Postcondition:

this.getExperimentID() = experiment

void setGatheredTime (const int & time)

Precondition:

ture

Postcondition:

this.getGatheredTime() = time
void setMarkedForDeletion (const bool value)

Parameters:

value Boolean, true if marked for deletion, false if not.

Precondition:

true

Postcondition:

this.getMarkedForDeletion() = value

void setName (const std::string & name)

Parameters:

name A string representing the name of the data.

Precondition:

true

Postcondition:

this.getName() = name /\ this.needs_storage = true

void setThumbnail (char * data, const int length)

Parameters:

data The thumbnail.

length An integer representing the length of the thumbnail.

Precondition:

true

Postcondition:

this.getThumbnailLength() = length /\ this.getDataItem() = data /\ this.needs_storage = true

void setThumbnailLength (const int & length)

Parameters:

length An integer representing the length of the thumbnail.

Precondition:

true

Postcondition:

this.getThumbnailLength() = length /\ this.needs_storage = true
void setType (const std::string & type)

Precondition:
    true

Postcondition:
    this.getType() = type

define setUpdate ()

Precondition:
    this.identifier > -1

Postcondition:
    this.needs_update = true

define store () throw (DataException)

Exceptions:

    DataException (p. 209)

Precondition:
    true

Postcondition:
    The data of the data object is stored into the database if the data is changed.

Member Data Documentation

    char* dataItem    [private]

    int dataItemLength [private]

    ExperimentID experiment  [private]

    int gatheredAt    [private]

    DataID identifier   [private]

    bool markedForDeletion [private]

    std::string name    [private]
bool needs_storage  [private]
Whether the data of this payload needs client-server communication to store its attributes.

bool needs_update  [private]
Whether the data of this payload needs client-server communication to update its attributes.

char* thumbnail  [private]

int thumbnailLength  [private]

std::string type  [private]
The documentation for this class was generated from the following files:

- common/data.h
- common/data.cpp

3.2.38  Data::DataException Class Reference

Exception class thrown by Data (p. 199).
#include <data.h>
Inherits runtime_error.
Inheritance diagram for Data::DataException:

Public Member Functions

- DataException (const std::string &arg)

Constructor & Destructor Documentation

DataException (const std::string & arg)  [inline]
The documentation for this class was generated from the following file:

- common/data.h
3.2.39 DatabaseAbstraction Class Reference

Type definition for SatelliteID.
#include <databaseabstraction.h>
Inherits AdminLowlevel.
Inheritance diagram for DatabaseAbstraction:

---

CHAPTER 3. COMPONENT DESCRIPTIONS

Inheritance diagram for DatabaseAbstraction:
CHAPTER 3. COMPONENT DESCRIPTIONS

Public Member Functions

- void accountAddExperiment (const AccountID &aid, const ExperimentID &eid) const
- void accountAddPayload (const AccountID &aid, const PayloadID &pid) const
- bool accountExists (AccountID id) const
- bool accountHasRightsToExperiment (const AccountID &aid, const ExperimentID &eid) const
- bool accountHasRightsToPayload (const AccountID &aid, const PayloadID &pid) const
- bool accountHasRightsToSatellite (const AccountID &aid, const SatelliteID &sid) const
- void accountRemoveExperiment (const AccountID &aid, const ExperimentID &eid) const
- void accountRemovePayload (const AccountID &aid, const PayloadID &pid) const
- void addLog (SingleLog aLog) const
- void connect () const
  
  Connect to the database.

- DatabaseAbstraction ()
  
  Default constructor.

- std::vector< BookingID > getAllBookingIDs (BookingType &type) const
- std::vector< CommandID > getAllCommandIDs () const
- std::vector< DataID > getAllDataIDs () const
- std::vector< ExperimentID > getAllExperimentIDs () const
- std::vector< PayloadID > getAllPayloadIDs () const
- std::vector< SatelliteID > getAllSatelliteIDs () const
- std::vector< SatelliteID > getAvailableSatellites (int time) const
- DataList getDataList (const ExperimentID &id) const
- ExperimentID getExperimentAtTime (const SatelliteID &satelliteID, const unsigned int time) const
- std::vector< BookingID > getExternalBookingsOfExperiment (ExperimentID id) const
- BookingID getFirstBooking () const
- BookingID getInternalBooking (SatelliteID satelliteID) const
- std::vector< ExperimentID > getPayLoadExperiments (const PayloadID &id) const
- std::vector< CommandID > getPayloadListOfCommands (const PayloadID &id) const
- std::list< QueueItem > getQueue (const SatelliteID &satelliteID, const BookingType &type) const
- DataList getThumbnailList (const ExperimentID &id) const
- std::vector< BookingID > getTimeBookingIDs (int time) const
- void removeExperiment (const ExperimentID &id) const
- void removePayload (const SatelliteID &satelliteID, const BookingType &type) const
- void removeQueueItem (const ItemID &id) const
- void retrieveAccount (const AccountID &id, AccountType &type, std::string &password, bool &registered, std::string &fullname, std::string &email) const
- void retrieveBooking (const BookingID &id, Direction &direction, BookingType &type, BookingStatus &status, int &startTime, int &endTime, ExperimentID &experimentID, GSSID &gssID) const
CHAPTER 3. COMPONENT DESCRIPTIONS

- void retrieveCommand (const CommandID &id, std::string &name, PayloadID &payload, CommandParam &commandParam) const
- void retrieveData (const DataID &id, std::string &name, ExperimentID &experiment, int &gatheredAt, std::string &type, char **dataItem, int &dataItemLength, char **thumbnail, int &thumbnailLength, bool &markedForDeletion) const
- void retrieveExperiment (const ExperimentID &id, std::string &name, PayloadID &payload, int &scheduledFrom, int &scheduledTo, ExperimentStatus &status, ListOfExperimentCommands &listOfPossibleCommands,ListOfCommands &listOfCommands) const
- SingleLog retrieveLog (LogID id) const
- std::vector<LogID> retrieveLogs (int starttime, int endtime, const SingleLog::Component &source, int priority, const ExperimentID &anExperimentID, const PayloadID &aPayloadID, const SatelliteID &aSatelliteID) const
- void retrievePayload (const PayloadID &id, std::string &name, bool &registered, SatelliteID &satelliteID) const
- void retrieveQueueItem (const ItemID &id, Action &action, SatelliteID &satID, PayloadID &payloadID, int &intVal, std::string &strVal, BookingType &type, bool &history) const
- void retrieveSatellite (const SatelliteID &id, std::string &name, bool &registered, std::string &mccUrl, std::string &mcsUrl) const
- void updateAccount (AccountID &aid, const AccountType &type, const std::string &password, const bool registered, const std::string &fullname, const std::string &email) const
- void updateBooking (BookingID &id, const Direction &direction, const BookingType &type, const BookingStatus &status, const int startTime, const int endTime, const ExperimentID &experimentID, const GSSID &gssID) const
- void updateCommand (CommandID &id, const std::string &name, const PayloadID &payload, const CommandParam &commandParam) const
- void updateData (DataID &id, const std::string &name, const ExperimentID &experiment, const int &gatheredAt, const std::string &type, const char *dataItem, const int &dataItemLength, const char *thumbnail, const int &thumbnailLength, const bool markedForDeletion) const
- void updateExperiment (ExperimentID &id, const std::string &name, const PayloadID &payload, const int &scheduledFrom, const int &scheduledTo, const ExperimentStatus &status, const ListOfExperimentCommands &listOfPossibleCommands) const
- void updatePayload (PayloadID &id, const std::string &name, const bool registered, const SatelliteID &satelliteID) const
- void updateQueueItem (ItemID &id, const Action &action, const SatelliteID &satID, const PayloadID &payloadID, const int intVal, const std::string &strVal, const BookingType &type, const bool history) const
- void updateSatellite (SatelliteID &id, const std::string &name, const bool registered, const std::string &mccUrl, const std::string &mcsUrl) const
- ~DatabaseAbstraction ()

Default destructor.

Private Member Functions

- void checkAccountIsRegistered (const AccountID &aid) const

Check whether an account is registered.
• void checkResult (mysqlpp::Result &result) const
  Checks whether the size of result is 1.

• template<class T>
  T executeAndProcessQuery (mysqlpp::Query &query) const
  Executes query and processes the result to be of type T.

• void executeQuery (mysqlpp::Query &query) const
  Convenience function to execute query.

• mysqlpp::Query newQuery () const
  Convenience function to create a new query object.

• mysqlpp::Result storeQuery (mysqlpp::Query &query) const
  Convenience function to execute query which can return a result.

Private Attributes

• mysqlpp::Connection connection
  Database connection.

• std::string db
  Database name.

• std::string host
  Host of the database server.

• std::string passwd
  Database password.

• std::string user
  Database user name.

Static Private Attributes

• static const int numberOfRetries = 10
  The number of retries in case the database connection fails.

Classes

• class DatabaseException

Detailed Description

Implementation class for AdminLowlevel (p. 126) using a MySQL database.
CHAPTER 3. COMPONENT DESCRIPTIONS

Constructor & Destructor Documentation

DatabaseAbstraction ()

~DatabaseAbstraction () [inline]

Member Function Documentation

void accountAddExperiment (const AccountID & aid, const ExperimentID & eid) const [virtual]

Implements AdminLowlevel (p. 132).

void accountAddPayload (const AccountID & aid, const PayloadID & pid) const [virtual]

Implements AdminLowlevel (p. 132).

bool accountExists (AccountID id) const [virtual]

Implements AdminLowlevel (p. 132).

bool accountHasRightsToExperiment (const AccountID & aid, const ExperimentID & eid) const [virtual]

Implements AdminLowlevel (p. 133).

bool accountHasRightsToPayload (const AccountID & aid, const PayloadID & pid) const [virtual]

Implements AdminLowlevel (p. 133).

bool accountHasRightsToSatellite (const AccountID & aid, const SatelliteID & sid) const [virtual]

Implements AdminLowlevel (p. 133).

void accountRemoveExperiment (const AccountID & aid, const ExperimentID & eid) const [virtual]

Implements AdminLowlevel (p. 134).

void accountRemovePayload (const AccountID & aid, const PayloadID & pid) const [virtual]

Implements AdminLowlevel (p. 134).

void addLog (SingleLog aLog) const [virtual]

Implements AdminLowlevel (p. 134).
void checkAccountIsRegistered (const AccountID & aid) const [private]

void checkResult (mysqlpp::Result & result) const [inline, private]

void connect () const

Postcondition:
connection is a valid database connection.

T executeAndProcessQuery (mysqlpp::Query & query) const [inline, private]

void executeQuery (mysqlpp::Query & query) const [private]

std::vector< BookingID > getAllBookingIDs (BookingType & type) const [virtual]

Implements AdminLowlevel (p. 135).

vector< CommandID > getAllCommandIDs () const [virtual]
Implements AdminLowlevel (p. 135).

vector< DataID > getAllDataIDs () const [virtual]
Implements AdminLowlevel (p. 135).

vector< ExperimentID > getAllExperimentIDs () const [virtual]
Implements AdminLowlevel (p. 135).

vector< PayloadID > getAllPayloadIDs () const [virtual]
Implements AdminLowlevel (p. 135).

vector< SatelliteID > getAllSatelliteIDs () const [virtual]
Implements AdminLowlevel (p. 136).

std::vector< SatelliteID > getAvailableSatellites (int time) const [virtual]
Implements AdminLowlevel (p. 136).

DataList getDataList (const ExperimentID & id) const [virtual]
Implements AdminLowlevel (p. 136).

ExperimentID getExperimentAtTime (const SatelliteID & satelliteID, const unsigned int time) const [virtual]
Implements AdminLowlevel (p. 136).
CHAPTER 3. COMPONENT DESCRIPTIONS

std::vector< BookingID > getExternalBookingsOfExperiment (ExperimentID id) const [virtual]
Implements AdminLowlevel (p. 137).

BookingID getFirstBooking () const [virtual]
Implements AdminLowlevel (p. 137).

BookingID getInternalBooking (SatelliteID satelliteID) const [virtual]
Implements AdminLowlevel (p. 137).

vector< ExperimentID > getPayloadExperiments (const PayloadID & id) const [virtual]
Implements AdminLowlevel (p. 137).

vector< CommandID > getPayloadListOfCommands (const PayloadID & id) const [virtual]
Implements AdminLowlevel (p. 138).

std::list< QueueItem > getQueue (const SatelliteID & satelliteID, const BookingType & type) const [virtual]
Implements AdminLowlevel (p. 138).

DataList getThumbnailList (const ExperimentID & id) const [virtual]
Implements AdminLowlevel (p. 138).

std::vector< BookingID > getTimeBookingIDs (int time) const [virtual]
Implements AdminLowlevel (p. 139).

mysqlpp::Query newQuery () const [private]
Todo
Try to clean up this code as it should not be needed for the above logic it merely fixes a warning

void removeExperiment (const ExperimentID & id) const [virtual]
Todo
Implement removal of experiment by simply executing an appropriate query.
Implements AdminLowlevel (p. 139).
CHAPTER 3. COMPONENT DESCRIPTIONS

void removeQueue (const SatelliteID & satelliteID, const BookingType & type) const [virtual]
Implements AdminLowlevel (p. 139).

void removeQueueItem (const ItemID & id) const [virtual]
Implements AdminLowlevel (p. 139).

void retrieveAccount (const AccountID & id, AccountType & type, std::string & password, bool & registered, std::string & fullname, std::string & email) const [virtual]
Implements AdminLowlevel (p. 140).

void retrieveBooking (const BookingID & id, Direction & direction, BookingType & type, BookingStatus & status, int & startTime, int & endTime, ExperimentID & experimentID, GSSID & gssID) const [virtual]
Implements AdminLowlevel (p. 140).

void retrieveCommand (const CommandID & id, std::string & name, PayloadID & payload, CommandParam & commandParam) const [virtual]
Implements AdminLowlevel (p. 141).

void retrieveData (const DataID & id, std::string & name, ExperimentID & experiment, int & gatheredAt, std::string & type, char ** dataItem, int & dataItemLength, char ** thumbnail, int & thumbnailLength, bool & markedForDeletion) const [virtual]
Todo
Implement retrieval of high resolution data
Implements AdminLowlevel (p. 141).

void retrieveExperiment (const ExperimentID & id, std::string & name, PayloadID & payload, int & scheduledFrom, int & scheduledTo, ExperimentStatus & status, ListOfExperimentCommands & listOfPossibleCommands, ListOfCommands & listOfCommands) const [virtual]
Implements AdminLowlevel (p. 142).

SingleLog retrieveLog (LogID id) const [virtual]
Implements AdminLowlevel (p. 142).

std::vector< LogID > retrieveLogs (int begintime, int endtime, const SingleLog::Component & source, int priority, const ExperimentID & anExperimentID, const PayloadID & aPayloadID, const SatelliteID & aSatelliteID) const [virtual]
Implements AdminLowlevel (p. 142).
void retrievePayload (const PayloadID & id, std::string & name, bool & registered, SatelliteID & satelliteID) const [virtual]

Implements AdminLowlevel (p. 143).

void retrieveQueueItem (const ItemID & id, Action & action, SatelliteID & satID, PayloadID & payloadID, int & intVal, std::string & strVal, BookingType & type, bool & history) const [virtual]

Implements AdminLowlevel (p. 143).

void retrieveSatellite (const SatelliteID & id, std::string & name, bool & registered, std::string & mccUrl, std::string & mcsUrl) const [virtual]

Implements AdminLowlevel (p. 144).

mysqlpp::Result storeQuery (mysqlpp::Query & query) const [private]

void updateAccount (AccountID & aid, const AccountType & type, const std::string & password, const bool registered, const std::string & fullname, const std::string & email) const [virtual]

Implements AdminLowlevel (p. 144).

void updateBooking (BookingID & id, const Direction & direction, const BookingType & type, const BookingStatus & status, const int startTime, const int endTime, const ExperimentID & experimentID, const GSSID & gssID) const [virtual]

Implements AdminLowlevel (p. 145).

void updateCommand (CommandID & id, const std::string & name, const PayloadID & payload, const CommandParam & commandParam) const [virtual]

Implements AdminLowlevel (p. 145).

void updateData (DataID & id, const std::string & name, const ExperimentID & experiment, const int & gatheredAt, const std::string & type, const char * dataItem, const int & dataItemLength, const char * thumbnail, const int & thumbnailLength, const bool markedForDeletion) const [virtual]

Todo

  implement saving dataItem

Todo

  Implement update query

Implements AdminLowlevel (p. 146).
void updateExperiment (ExperimentID & id, const std::string & name, const PayloadID & payload, const int & scheduledFrom, const int & scheduledTo, const ExperimentStatus & status, constListOfExperimentCommands & listOfPossibleCommands) const [virtual]
Implements AdminLowlevel (p. 146).

void updatePayload (PayloadID & id, const std::string & name, const bool registered, const SatelliteID & satelliteID) const [virtual]
Implements AdminLowlevel (p. 147).

void updateQueueItem (ItemID & id, const Action & action, const SatelliteID & satID, const PayloadID & payloadID, const int intVal, const std::string & strVal, const BookingType & type, const bool history) const [virtual]
Implements AdminLowlevel (p. 147).

void updateSatellite (SatelliteID & id, const std::string & name, const bool registered, const std::string & mccUrl, const std::string & mcsUrl) const [virtual]
Implements AdminLowlevel (p. 147).

Member Data Documentation

mysqlpp::Connection connection [mutable, private]

std::string db [private]

std::string host [private]

const int numberOfRetries = 10 [static, private]

std::string passwd [private]

std::string user [private]

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

- server/databaseabstraction.h
- server/databaseabstraction.cpp

3.2.40 DatabaseAbstraction::DatabaseException Class Reference

#include <databaseabstraction.h>

Inherits AdminLowlevel::AdminLowlevelException.
Inheritance diagram for DatabaseAbstraction::DatabaseException:

```
DatabaseAbstraction::DatabaseException
    + DatabaseException()

AdminLowlevelException
    + AdminLowlevelException()

runtime_error
    + DatabaseException()
```

Public Member Functions

- **DatabaseException** (const std::string &arg)

Detailed Description

Exception class thrown by DatabaseAbstraction (p. 210).

Constructor & Destructor Documentation

**DatabaseException (const std::string & arg)** [inline]

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

- server/databaseabstraction.h

3.2.41 Experiment Class Reference

Include `<experiment.h>`

Public Member Functions

- void **addCommand** (const CommandID &command, const ParamTypes &currentValue, const int &time) throw (ExperimentException)
  
  *Add command with parameters with time to the experiment.*

- **Experiment** (const ExperimentID id) throw (ExperimentException)
  
  *Constructor of a new experiment object with ExperimentID id.*

- **Experiment** (const std::string &name, const PayloadID &payload)
  
  *Constructor of a new experiment object, which is not yet in the database.*
• **Experiment ()**
  
  Empty constructor.

• **DataList getDataList () const throw (ExperimentException)**
  
  Returns the list of data for the experiment.

• **ExperimentID getID ()**
  
  Returns the identifier for this experiment.

• **ListOfExperimentCommands getListOfCommands () throw (ExperimentException)**
  
  Returns the list of commands that currently make up the experiment.

• **ListOfCommands getListOfPossibleCommands () throw (ExperimentException)**
  
  Returns identifiers of all commands applicable this experiment.

• **std::string getName () throw (ExperimentException)**
  
  Returns the name of this experiment.

• **PayloadID getPayloadID () throw (ExperimentException)**
  
  Returns the payload to which the experiment belongs to.

• **std::pair<int, int> getSchedule ()**
  
  Returns the current schedule of the experiment.

• **ExperimentStatus getStatus () throw (ExperimentException)**
  
  Returns the status of this experiment.

• **DataList getThumbnailList () const throw (ExperimentException)**
  
  Returns the list of thumbnails for the experiment.

• **void instantiate () throw (ExperimentException)**
  
  Retrieves data from experiment with identifier from lower layer.

• **void modifyCommand (unsigned int i, const ParamTypes &newValue, const int &newTime) throw (ExperimentException)**
  
  Modify the current value and the time of the command with index i.

• **void removeCommand (unsigned int i) throw (ExperimentException)**
  
  Removes command with index i from the experiment.

• **void removeExperiment () throw (ExperimentException)**
  
  Removes the experiment.

• **void schedule (const int &startTime, const int &endTime) throw (ExperimentException)**
  
  A request to schedule the experiment is made.

• **void setListOfCommands (const ListOfExperimentCommands &commands)**
  
  Sets the list of possible commands to commands.
CHAPTER 3. COMPONENT DESCRIPTIONS

- void **setName** (const std::string &newName)
  
  Sets the name of this experiment.

- void **setPayloadID** (const PayloadID &payload)
  
  Sets the payload to which the experiment belongs to.

- void **setScheduledTo** (int endTime)
  
  Sets the relative end time of the experiment.

- void **setStatus** (const ExperimentStatus &status)
  
  Sets the status of this experiment.

- void **setUpdate** ()
  
  Indicate that the data needs to be updated.

- void **store** () throw (ExperimentException)
  
  Stores the data of the payload into the database.

- void **unschedule** ()
  
  A request to unschedule the experiment is made.

- ~**Experiment** ()
  
  Destructor of experiment object.

Private Attributes

- ExperimentID **identifier**
  
  Identifier for this object in the lower layer.

- ListOfExperimentCommands **listOfCommands**
  
  List of commands of the experiment.

- ListOfCommands **listOfPossibleCommands**
  
  List of possible commands for the experiment.

- std::string **name**
  
  Name of this experiment.

- bool **needs_storage**
- bool **needs_update**
- PayloadID **payload**
  
  Payload (p. 276) to which the experiment belongs to.

- int **scheduledFrom**
  
  Start time of schedule.

- int **scheduledTo**
  
  End time of schedule.
• **ExperimentStatus status**

  Status of the experiment.

**Classes**

  • class **ExperimentException**

  **Constructor & Destructor Documentation**

  **Experiment ()**

  **Precondition:**

  true

  **Postcondition:**

  this.identifier = -1 \ this.name = "" \ this.status = local \ this.payload = -1 \ this.scheduledFrom = -1 \ this.scheduledTo = -1

  **Experiment (const std::string & name, const PayloadID & payload)**

  **Parameters:**

  name The name of the experiment.

  payload The payload to which the experiment belongs to.

  **Precondition:**

  true

  **Postcondition:**

  this.name = name \ this.payload = payload \ this.identifier = -1 \ this.status = local \ this.scheduledFrom = -1 \ this.scheduledTo = -1

  **Experiment (const ExperimentID id) throw (ExperimentException)**

  **Exceptions:**

  *ExperimentException* (p. 231)

  **Parameters:**

  id an ExperimentID identifying the satellite in the lower layer.

  **Precondition:**

  id >= 0 \ (exists e in conn.experiment ( e.id = id))

  **Postcondition:**

  this.getID() = id
~Experiment ()

Precondition:
  true

Member Function Documentation

void addCommand (const CommandID & command, const ParamTypes & currentValue, const int & time) throw (ExperimentException)

Exceptions:
  ExperimentException (p. 231)

Parameters:
  command a CommandID identifying a command.
  currentValue a ParamTypes (p. 275) representing the value of the command.
  time the (relative) time at which the command will be scheduled.

Precondition:
  cp = conn.getCommandParameters(command) /\ cp.minValue <= currentValue <= cp.maxValue

Postcondition:
  (command, value, time) in getListOfCommands() (p. 226)

DataList getDataList () const throw (ExperimentException)

Exceptions:
  ExperimentException (p. 231)

Precondition:
  true

Returns:
  this.dataList

ExperimentIDgetID ()

Precondition:
  true

Returns:
  identifier
ListOfExperimentCommands getListOfCommands () throw (ExperimentException)

Exceptions:

*ExperimentException* (p. 231)

Precondition:

true

Returns:

this.listOfCommands

ListOfCommands getListOfPossibleCommands () throw (ExperimentException)

Exceptions:

*ExperimentException* (p. 231)

Precondition:

this->payload > 0

Returns:

\{c.getID() | (c in connection.Command) \&\& (c.getPayload() = this.getPayload())\}

std::string getName () throw (ExperimentException)

Exceptions:

*ExperimentException* (p. 231)

Precondition:

true

Returns:

this.name

PayloadID getPayloadID () throw (ExperimentException)

Exceptions:

*ExperimentException* (p. 231)

Precondition:

true

Returns:

this.payload
std::pair<int, int> getSchedule ()

Precondition:
true

Returns:
(this.scheduleFrom, this.scheduledTo)

ExperimentStatus getStatus () throw (ExperimentException)

Exceptions:
ExperimentException (p. 231)

Precondition:
true

Returns:
this.status

DataList getThumbnailList () const throw (ExperimentException)

Exceptions:
ExperimentException (p. 231)

Precondition:
true

Returns:
this.thumbnailList

void instantiate () throw (ExperimentException)

Exceptions:
ExperimentException (p. 231)

Precondition:
identifier >= 0

Postcondition:
The parameters of experiment are instantiated with information from the lower layer.
void modifyCommand (unsigned int i, const ParamTypes & newValue, const int & newTime) throw (ExperimentException)

Exceptions:

   ExperimentException (p. 231)

Parameters:

   i  an index in the vector.
   newValue a ParamTypes (p. 275) which represents the new value of the command.
   newTime a time which represents the new time of the command.

Precondition:

   0 <= i < length(listOfCommands)

Postcondition:

   listOfCommands[i].currentValue = newValue /
   listOfCommands[i].time = newTime


void removeCommand (unsigned int i) throw (ExperimentException)

Exceptions:

   ExperimentException (p. 231)

Precondition:

   0 <= i < length(listOfCommands)

Postcondition:

   Command (p. 162) with index i removed from dataList.


void removeExperiment () throw (ExperimentException)

Exceptions:

   ExperimentException (p. 231)

Precondition:

   true

Postcondition:

   not (exists e in conn.experiment ( e = this ) )
void schedule (const int & startTime, const int & endTime) throw (ExperimentException)

Exceptions:

\[ \text{ExperimentException (p. 231)} \]

Parameters:

\[ \text{startTime} \] the time at which the experiment should start.
\[ \text{endTime} \] the time at which the experiment should end.

Precondition:

\[ \text{startTime} \leq \text{endTime} \]

Postcondition:

\[ \text{this.scheduledFrom} = \text{startTime} /\backslash \text{this.scheduledTo} = \text{endTime} \]

void setListOfCommands (const ListOfExperimentCommands & commands)

Parameters:

\[ \text{commands} \] The new list of possible commands of the experiment.

Precondition:

true

Postcondition:

\[ \text{this.getListOfPossibleCommands}() = \text{commands} \]

void setName (const std::string & newName)

Parameters:

\[ \text{newName} \] The new name of the experiment.

Precondition:

true

Postcondition:

\[ \text{this.getName}() = \text{newName} \]

void setPayloadID (const PayloadID & payload)

Parameters:

\[ \text{payload} \] a payloadID representing the payload the experiment belongs to.
Precondition:
    true

Postcondition:
    this.getPayload() = payload

void setScheduledTo (int \textit{endTime})

Parameters:
    \textit{endTime} An integer telling how long the experiment takes, in seconds.

Precondition:
    endTime >= 0

Postcondition:
    this.scheduledTo = endTime

void setStatus (const ExperimentStatus & \textit{status})

Parameters:
    \textit{status} The new status of the experiment.

Precondition:
    true

Postcondition:
    this.getStatus() = status

void setUpdate ()

Precondition:
    this.identifier > -1

Postcondition:
    this.needs_update = true

void store () throw (ExperimentException)

Exceptions:
    \textit{ExperimentException} (p. 231)

Precondition:
    true

Postcondition:
    The data of the payload is stored into the database if the data is changed.
void unschedule ()

Precondition:
    true

Postcondition:
    this.scheduledFrom = -1 /\ this.scheduledTo = -1

Member Data Documentation

ExperimentID identifier [private]

ListOfExperimentCommands listOfCommands [private]

ListOfCommands listOfPossibleCommands [private]

std::string name [private]

bool needs_storage [private]
Whether the data of this payload needs client-server communication to store its attributes.

bool needs_update [private]
Whether the data of this payload needs client-server communication to update its attributes.

PayloadID payload [private]

int scheduledFrom [private]

int scheduledTo [private]

ExperimentStatus status [private]
The documentation for this class was generated from the following files:

- common/experiment.h
- common/experiment.cpp

3.2.42 Experiment::ExperimentException Class Reference

#include <experiment.h>
Inherits runtime_error.
CHAPTER 3. COMPONENT DESCRIPTIONS

Inheritance diagram for Experiment::ExperimentException:

```
+ runtime_error

<table>
<thead>
<tr>
<th>Experiment::ExperimentException</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ ExperimentException()</td>
</tr>
</tbody>
</table>
```

Public Member Functions

- **ExperimentException** (const std::string &arg)

Constructor & Destructor Documentation

**ExperimentException (const std::string & arg)** [inline]

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

```
common/experiment.h
```

3.2.43 ExperimentCommand Class Reference

Parameters of a command as used in an experiment.

```
#include <defs.h>
```

Public Member Functions

- **bool operator==(const ExperimentCommand &other) const**

Public Attributes

- **CommandID command**
  
  *Identifies the command.*

- **ParamTypes currentValue**
  
  *Identifies the current value of the command.*

- **ParamType paramType**
  
  *Type of the parameter.*

- **int time**
  
  *Identifies the time of execution.*
CHAPTER 3. COMPONENT DESCRIPTIONS

Member Function Documentation

bool operator==(const ExperimentCommand & other) const [inline] < Needed for testing.

Member Data Documentation

CommandID command

ParamTypes currentValue

ParamType paramType

int time
   The documentation for this class was generated from the following file:
   
   • common/defs.h

3.2.44 ExperimentPart Class Reference

Part of an experiment, as used in on MCC/MCS and scheduler.
#include <defs.h>

Public Attributes

• std::string command
• int time

Member Data Documentation

std::string command

int time
   The documentation for this class was generated from the following file:
   
   • common/defs.h

3.2.45 failed Class Reference

Inherits xmlrpc_c::method.
Inheritance diagram for failed:

```
method

failed
- controller
- sat
+ execute()
+ failed()
```

Public Member Functions

- void `execute` (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
- failed (Satellite *tmpSat, McsController *tmpController)

Private Attributes

- McsController * controller
- Satellite * sat

Constructor & Destructor Documentation

failed (Satellite * tmpSat, McsController * tmpController) [inline]

Member Function Documentation

void `execute` (xmlrpc_c::paramList const & paramList, xmlrpc_c::value * const retvalP) [inline]

Member Data Documentation

McsController* controller [private]

Satellite* sat [private]

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

- server/mcsListener.cpp

3.2.46 getChallengeAndSession Class Reference

Returns a challenge and a session, to be used by the user to identify himself to the server.

#include <clientconnectivity.h>
CHAPTER 3. COMPONENT DESCRIPTIONS

Inherits xmlrpc_c::method.

Inheritance diagram for getChallengeAndSession:

```
    method
       ^
      getChallengeAndSession
            - controller
            + execute()
            + getChallengeAndSession()
```

Public Member Functions

- void `execute` (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
- `getChallengeAndSession` (Controller *controller)

Private Attributes

- Controller * controller

Detailed Description

Precondition:

```
true
```

Returns:

A new session to be used for login (p. 247) and a challenge.

Constructor & Destructor Documentation

`getChallengeAndSession` (Controller * controller)

Member Function Documentation

```
void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value *const retvalP)
```

Member Data Documentation

Controller* controller [private]

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

```
- server/clientconnectivity.h
- server/clientconnectivity.cpp
```
### 3.2.47 getExperiment Class Reference

Get experiment by identifier.

```cpp
#include <clientconnectivity.h>
```

Inherits xmlrpc_c::method.

Inheritance diagram for getExperiment:

```
method

getExperiment
- controller
+ execute()
+ getExperiment()
```

#### Public Member Functions

- `void execute(xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)`
- `getExperiment(Controller *tmpController)`

#### Private Attributes

- `Controller * controller`

#### Detailed Description

**Parameters:**

- `sessionID` The first parameter, identifying the requesting session.
- `expID` The second parameter, an identifier of the experiment for which information should be retrieved.

**Precondition:**

`expID >= 0`

**Returns:**

An xmlRpc struct with the fields of the experiment with expID.

#### Constructor & Destructor Documentation

`getExperiment (Controller * tmpController)`

#### Member Function Documentation

`void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value * const retvalP)`
CHAPTER 3. COMPONENT DESCRIPTIONS

Member Data Documentation

Controller* controller [private]

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

- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.48 getExperiments Class Reference

Gets all satellites, payloads and experiments associated with an account.

#include <clientconnectivity.h>

Inherits xmlrpc_c::method.

Inheritance diagram for getExperiments:

```
method

getExperiments
  - controller
  + execute()
  + getExperiments()
```

Public Member Functions

- void execute (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
- getExperiments (Controller *tmpController)

Private Attributes

- Controller * controller

Detailed Description

Parameters:

sessionID The first parameter, identifying the requesting session and account.

Precondition:

true (??)

Returns:

[(Satellite (p. 293), [(Payload (p. 276), [Experiment (p. 221)])])], where Experiments are associated with the Payload (p. 276), the Payload (p. 276) is associated with the Satellite (p. 293) and the Satellite (p. 293) is associated with the requesting account.
CHAPTER 3. COMPONENT DESCRIPTIONS

Constructor & Destructor Documentation

getExperiments (Controller * tmpController)

Member Function Documentation

void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value *const retvalP)

Member Data Documentation

Controller* controller [private]
The documentation for this class was generated from the following files:

- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.49 getListOfOpportunities Class Reference

Returns a list of opportunities for an experiment with experimentID.

#include <clientconnectivity.h>

Inherits xmlrpc_c::method.

Inheritance diagram for getListOfOpportunities:

Public Member Functions

- void execute (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
- getListOfOpportunities (Controller *tmpController)

Private Attributes

- Controller * controller

Detailed Description

Parameters:

session The first parameter, identifying the session requesting the list of opportunities.
CHAPTER 3. COMPONENT DESCRIPTIONS

*experimentID* The second parameter, identifying the experiment for which the list of opportunities is requested.

*start* Start time of the interval, opportunities in which should be returned.

*end* End of the interval.

Precondition:

\[
\text{experimentID} \geq 0
\]

Returns:

\[
[b: \text{BookingInfo} (p. 157) | b \text{ belongs to } s \text{ with } s.\text{identifier} = \text{SatelliteID} /\ start < b.\text{start} /\ b.\text{end} < end]
\]

Constructor & Destructor Documentation

`getListOfOpportunities (Controller * tmpController)`

Member Function Documentation

`void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value *const retvalP)`

Member Data Documentation

`Controller* controller [private]`

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

- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.50 `getListOfThumbnails` Class Reference

Requests a list of thumbnails.

#include `<clientconnectivity.h>`

Inherits `xmlrpc_c::method`.

Inheritance diagram for `getListOfThumbnails`:
Public Member Functions

- void \texttt{execute} (xmlrpc\_c::paramList const \&paramList, xmlrpc\_c::value *const retvalP)
- \texttt{getListOfThumbnails} (Controller *tmpController)

Private Attributes

- Controller * controller

Detailed Description

Parameters:

- \texttt{session} The session requesting the unschedule.
- \texttt{id} The experiment id for which thumbnails are requested.

Returns:

The list of thumbnails belonging to ID.

Constructor & Destructor Documentation

\texttt{getListOfThumbnails} (Controller * \texttt{tmpController})

Member Function Documentation

\texttt{void execute} (xmlrpc\_c::paramList const & \texttt{paramList}, xmlrpc\_c::value *const \texttt{retvalP})

Member Data Documentation

\texttt{Controller* controller} [private]

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

- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.51 getPayload Class Reference

Get payload by identifier.

\texttt{#include <clientconnectivity.h>}

Inherits xmlrpc\_c::method.
CHAPTER 3. COMPONENT DESCRIPTIONS

Inheritance diagram for getPayload:

```
method

getPayload
- controller
+ execute()
+ getPayload()
```

Public Member Functions

- void **execute** (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
- getPayload (Controller *tmpController)

Private Attributes

- Controller * controller

Detailed Description

Parameters:

- **sessionID** The first parameter, identifying the requesting session.
- **payID** The second parameter, an identifier of the payload for which information should be retrieved.

Precondition:

payID >= 0

Returns:

An xmlRpc struct with the fields of the payload with payID.

Constructor & Destructor Documentation

getPayload (Controller * tmpController)

Member Function Documentation

void **execute** (xmlrpc_c::paramList const & **paramList**, xmlrpc_c::value *const **retvalP**)

Member Data Documentation

**Controller** * controller  **[private]**

The documentation for this class was generated from the following files:
3.2.52 getSatellite Class Reference

Get satellite by identifier.

```cpp
#include <clientconnectivity.h>

Inherits xmlrpc_c::method.
```

Inheritance diagram for getSatellite:

Public Member Functions

- void `execute` (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
- getSatellite (Controller *tmpController)

Private Attributes

- Controller * controller

Detailed Description

Parameters:

- `sessionID` The first parameter, identifying the current session.
- `satID` The second parameter. An identifier of the satellite of which information should be retrieved.

Precondition:

- `satID` >= 0

Returns:

- An xmlRpc struct with the fields of the satellite with `satID`.

Constructor & Destructor Documentation

getSatellite (Controller * `tmpController`)
CHAPTER 3. COMPONENT DESCRIPTIONS

Member Function Documentation

void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value * const retvalP)

Member Data Documentation

Controller* controller [private]
The documentation for this class was generated from the following files:

- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.53 getThumbnail Class Reference

Get the next observation thumbnail for an experiment.

#include <clientconnectivity.h>

Inherits xmlrpc_c::method.

Inheritance diagram for getThumbnail:

```
method
```

```
getThumbnail
- controller
+ execute()
+ getThumbnail()
```

Public Member Functions

- void execute (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
- getThumbnail (Controller *controller)

Private Attributes

- Controller * controller

Detailed Description

Parameters:

- **session** The session requesting the thumbnail.

- **experimentID** The identifier of the experiment for which the thumbnail is requested.
Precondition:

\[ \text{experimentID} \geq 0 \]

Returns:

The next thumbnail for the experiment with experimentID.

Constructor & Destructor Documentation

\textit{getThumbnail (Controller * controller)}

Member Function Documentation

\texttt{void execute (xmlrpc\_c::paramList const & paramList, xmlrpc\_c::value *const retvalP)}

Member Data Documentation

\texttt{Controller* controller [private]}

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

- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.54 Interval Class Reference

Specifies an interval in time, to be used in a booking.

\texttt{#include <defs.h>}

Public Member Functions

- \texttt{bool operator== (const Interval &other) const}

Public Attributes

- \texttt{int end}
  \textit{The end time of the interval.}
- \texttt{int start}
  \textit{The start time of the interval.}

Member Function Documentation

\texttt{bool operator== (const Interval & other) const [inline]}

\texttt{< Needed for testing}
Member Data Documentation

int end

int start

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

- common/defs.h

3.2.55 listFailure Class Reference

Inherits xmlrpc_c::method.

Inheritance diagram for listFailure:

```
method

listFailure

- controller
- sat
+ execute()
+ listFailure()
```

Public Member Functions

- void execute (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
- listFailure (Satellite *tmpSat, MccController *tmpController)

Private Attributes

- MccController * controller
- Satellite * sat

Constructor & Destructor Documentation

listFailure (Satellite * tmpSat, MccController * tmpController) [inline]

Member Function Documentation

void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value *const retvalP) [inline]
Member Data Documentation

MccController* controller [private]

Satellite* sat [private]

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

- server/mccListener.cpp

3.2.56 listOfOpportunities Class Reference

Inherits xmlrpc_c::method.

Inheritance diagram for listOfOpportunities:

Public Member Functions

- void execute (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
- listOfOpportunities (Satellite *tmpSat, MccController *tmpController)

Private Attributes

- MccController * controller
- Satellite * sat

Constructor & Destructor Documentation

listOfOpportunities (Satellite *tmpSat, MccController *tmpController) [inline]

Member Function Documentation

void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value *const retvalP) [inline]

Member Data Documentation

MccController* controller [private]
CHAPTER 3. COMPONENT DESCRIPTIONS

Satellite* sat [private]
The documentation for this class was generated from the following file:

- server/mccListener.cpp

3.2.57 LogException Class Reference

< std::exception
#include <log.h>
Inherits runtime_error.
Inheritance diagram for LogException:

Public Member Functions

- LogException (const std::string &arg)

Detailed Description

< std::vector

Constructor & Destructor Documentation

LogException (const std::string & arg) [inline]
The documentation for this class was generated from the following file:

- common/log.h

3.2.58 login Class Reference

Login remote procedure call.
#include <clientconnectivity.h>
Inherits xmlrpc_c::method.

DDD 0.1 247
Inheritance diagram for login:

```
method
  login
  - controller
    + execute()
    + login()
```

Public Member Functions

- void `execute` (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
- `login` (Controller *tmpController)

Private Attributes

- Controller * controller

Detailed Description

Parameters:

- `session` The first parameter, the session the user has requested
- `name` The second parameter in the call, the name of the user to be logged in.
- `password` The third parameter in the call, the password given by name.

Precondition:

true

Postcondition:

If session and password are correct, the user is logged in. Otherwise, he is not.

Returns:

Whether the `login` (p. 247) succeeded.

Constructor & Destructor Documentation

`login` (Controller * `tmpController`)

Member Function Documentation

void `execute` (xmlrpc_c::paramList const & paramList, xmlrpc_c::value *const `retvalP`)
Member Data Documentation

Controller* controller  [private]
The documentation for this class was generated from the following files:

- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.59  logout Class Reference

Logout remote procedure call.
#include <clientconnectivity.h>
Inherits xmlrpc_c::method.

Inheritance diagram for logout:

Public Member Functions

- void execute (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
- logout (Controller *tmpController)

Private Attributes

- Controller * controller

Detailed Description

Parameters:

sessionId The first parameter in the call, a string designating the session.
name The second parameter in the call, the name of the user to be logged out

Precondition:

SessionID belongs to name.

Postcondition:

The user is logged out if the sessionID is correct, else they are not.
CHAPTER 3. COMPONENT DESCRIPTIONS

Returns:

Whether the logout (p. 249) succeeded.

Constructor & Destructor Documentation

logout (Controller ∗ tmpController)

Member Function Documentation

void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value ∗const retvalP)

Member Data Documentation

Controller* controller [private]

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

- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.60 MccClient Class Reference

This class can be used to send messages to a MCC.

#include <mccClient.h>

Inherits XmlRpcClient.

Inheritance diagram for MccClient:

[Inheritance diagram image]

Public Member Functions

- int cancelBooking (const BookingInfo &booking) const
  
  Send a request to the MCC to cancel a given pass.

- int getListOfopportunities (const Interval &period) const
Send a request to the MCC for a list of opportunities.

- **MccClient** (const std::string &myUrl, const std::string &serverUrl)
  
  Constructor of MCC client object.

- **int requestBooking** (const BookingInfo &booking) const
  
  Send a request to the MCC to book a given pass.

Protected Attributes

- xmlrpc::carriageParm::curl * myCarriageParmPtr
  
  The parameters used for the connection.

- xmlrpc::client::xml * myClientPtr
  
  The actual XML-RPC client.

- std::string myUrl
  
  The URL that can be used to report.

Constructor & Destructor Documentation

MccClient (const std::string & myUrl, const std::string & serverUrl) [inline]

Parameters:

  - **myUrl** The local URL the server can use for his response.
  - **serverUrl** The URL for the server.

Precondition:

  true

Member Function Documentation

int cancelBooking (const BookingInfo & booking) const

Parameters:

  - **booking** The booking indicating the pass that has to be canceled.

Precondition:

  true

Returns:

  The id that will be used when the answer is send.
CHAPTER 3. COMPONENT DESCRIPTIONS

int getListOfopportunities (const Interval & period) const

Parameters:
   period The period for which you want to retrieve the opportunities.

Precondition:
   true

Returns:
   The id that will be used when the answer is send.

int requestBooking (const BookingInfo & booking) const

Parameters:
   booking The booking indicating the pass that has to be booked (p. 149).

Precondition:
   true

Returns:
   The id that will be used when the answer is send.

Member Data Documentation

xmlrpc_c::carriageParm_curl0* myCarriageParmPtr [protected, inherited]

xmlrpc_c::client_xml* myClientPtr [protected, inherited]

std::string myUrl [protected, inherited]
The documentation for this class was generated from the following files:

   - server/mccClient.h
   - server/mccClient.cpp

3.2.61 MccController Class Reference

This class manages all connections with the MCCs.
#include <mccController.h>
Inherits XmlRpcController.
CHAPTER 3. COMPONENT DESCRIPTIONS

Inheritance diagram for MccController:

```
MccController
# infoList
+ callbackBooking()
+ callbackList()
+ cancelBooking()
+ checkForTimeouts()
+ ... lastUsed
# scheduler
# timeouts
+ getHostname()
+ removeBooking()
+ ~XmlRpcController()
# getNewSocket()
```

Public Member Functions

- void `callbackBooking` (Satellite *sat, const int id, const BookingStatus &status, const BookingInfo &booking)
  
  Callback function used for replies to requestBooking.

- void `callbackList` (Satellite *sat, const int id, const bool status, const BookingList &list)
  
  Callback function used for replies to getListOfopportunities.

- bool `cancelBooking` (const int bookingId, Satellite &sat, const BookingInfo &booking)
  
  Send a request to the MCC to cancel a given booking.

- void `checkForTimeouts` ()

- std::string `getHostname` ()
  
  Retrieve the local hostname from the config file 'server.icf'.

- bool `getListOfOpportunities` (const int bookingId, Satellite &sat, const Interval &period)
  
  Send a request to the MCC for a list of opportunities.

- MccController (Scheduler *scheduler)
  
  Constructor for MccController (p. 252) object.

- int `removeBooking` (Satellite *sat, int id)
  
  Removes information of the request with identifier id.

- bool `requestBooking` (const int bookingId, Satellite &sat, const BookingInfo &booking)
  
  Send a request to the MCC to book a given pass.
Protected Member Functions

- void checkSatellite (Satellite &sat)
  Check if there is already a MCC client and listener for an satellite, and if not create them.

- SocketInfo getNewSocket ()
  A function to get a socket bound to an unused port.

Protected Attributes

- std::map< SatelliteID, std::map< int, int > > bookingIds
  A map used for mapping server ids to bookingIds per satellite.

- std::map< SatelliteID, MccInfo * > infoList
  A map for storing all XML RPC clients and listeners.

- int lastUsed
  The port that was last used for a XML RPC listener.

- Scheduler * scheduler
  Scheduler (p. 302) used for callbacks.

- std::map< int, int > timeouts
  A map used for mapping server bookingIds to timeouts per satellite.

Constructor & Destructor Documentation

MccController (Scheduler * scheduler)

Parameters:

  scheduler The scheduler used for callbacks.

Member Function Documentation

void callbackBooking (Satellite * sat, const int id, const BookingStatus & status, const BookingInfo & booking)

Parameters:

  sat The satellite ofr which the reply has been received.
  id The id of the reply.
  status The status of the booking.
  booking Info about the requested booking.

Precondition:

  true

Postcondition:

  The reply has been forwarded to the scheduler.
void callbackList (Satellite * sat, const int id, const bool status, const BookingList & list)

Parameters:
  sat The satellite for which the reply has been received.
  id The id of the reply.
  status Whether the request was successful.
  list The list of opportunities, is empty if there was an error.

Precondition:
  true

Postcondition:
  The reply has been forwarded to the scheduler.

bool cancelBooking (const int bookingId, Satellite & sat, const BookingInfo & booking)

Parameters:
  bookingId The id that has to be used for reporting the reply.
  sat The satellite to which the message has to be send.
  booking The booking indicating the pass that has to be cancelled (p. 158).

Precondition:
  true

Postcondition:
  bookingIds[sat.sid][id] = bookingId.

Returns:
  Whether the message has been send successful.

void checkForTimeouts ()

void checkSatellite (Satellite & sat) [protected]

Parameters:
  sat The satellite that has to be checked.

Precondition:
  true

Postcondition:
  A MCC client and listener for the given satellite exist.
string getHostname () [inherited]

bool getListOfOpportunities (const int bookingId, Satellite & sat, const Interval & period)

Parameters:

bookingId The id that has to be used for reporting the reply.
sat The satellite to which the message has to be send.
period The period for which you want to retrieve the opportunities.

Precondition:
true

Postcondition:
bookingIds[sat.sid][id] = bookingId.

Returns:
Whether the message has been send successful.

SocketInfo getNewSocket () [protected, inherited]

Precondition:
true

Returns:
A socket bound to a port and that port.

int removeBooking (Satellite * sat, int id) [inherited]

Parameters:

sat Satellite (p. 293) to which the request belongs.
id The identifier of the request.

bool requestBooking (const int bookingId, Satellite & sat, const BookingInfo & booking)

Parameters:

bookingId The id that has to be used for reporting the reply.
sat The satellite to which the message has to be send.
booking The booking indicating the pass that has to be booked (p. 149).

Precondition:
true
CHAPTER 3. COMPONENT DESCRIPTIONS

Postcondition:

bookingIds[sat.sid][id] = bookingId.

Returns:

Whether the message has been send successful.

Member Data Documentation

std::map<SatelliteID, std::map<int, int>> bookingIds [protected, inherited]

std::map<SatelliteID, MccInfo*> infoList [protected]

int lastUsed [protected, inherited]

Scheduler* scheduler [protected, inherited]

std::map<int, int> timeouts [protected, inherited]

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

- server/mccController.h
- server/mccController.cpp

3.2.62 MccInfo Class Reference

This class contains information for communication with a MCC.

#include <mccController.h>

Public Attributes

- MccClient * client
- int port
- pthread_t thread

Member Data Documentation

MccClient* client

int port

pthread_t thread

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

- server/mccController.h
3.2.63 MccListener Class Reference

This class can be used to receive messages from a MCC.

```cpp
#include <mccListener.h>
```

Inherits XmlRpcListener.

Inheritance diagram for MccListener:

```
+ MccListener()
+ MccListener()
+ ~MccListener()
```

**Public Member Functions**

- **MccListener ()**
  
  Empty constructor of MCC listener object.

- **MccListener (const int socket, Satellite &sat, MccController &controller)**
  
  Constructor of MCC listener object.

- **void run ()**
  
  Start listening for incoming messages.

- **~MccListener ()**
  
  Destructor of MCC listener object.

**Protected Attributes**

- **MccController * controller**
  
  Controller (p. 170) used to report incomming messages.

- **xmlrpc_c::serverAbyss * myAbyssServerPtr**
  
  The XML RPC server.

- **Satellite sat**
  
  The satellite for which messages are received.
CHAPTER 3. COMPONENT DESCRIPTIONS

Constructor & Destructor Documentation

MccListener (const int socket, Satellite & sat, MccController & controller)

Parameters:

socket The socket that has to be used for listening.
sat The satellite for which the listener listens.
controller The controller that has to be called when receiving a message

Precondition:
true

MccListener () [inline]

~MccListener ()

Member Function Documentation

void run () [inherited]

Precondition:
true

Postcondition:
The listeners is listening for incoming messages.

Member Data Documentation

MccController* controller [protected]

xmlrpc_c::serverAbyss* myAbyssServerPtr [protected, inherited]

Satellite sat [protected]
The documentation for this class was generated from the following files:

- server/mccListener.h
- server/mccListener.cpp

3.2.64 McsClient Class Reference

This class can be used to send messages to a MCS.

#include <mcsClient.h>

Inherits XmlRpcClient.
CHAPTER 3. COMPONENT DESCRIPTIONS

Inheritance diagram for McsClient:

```
McsClient
+ erase()
+ exec()
+ fetch()
+ kill()
+ list()
+ McsClient()
+ run()
+ schedule()
+ store()
```

```
XmlRpcClient
# myCarriageParmPtr
# myClientPtr
# myUrl
+ XmlRpcClient()
+ ~XmlRpcClient()
```

### Public Member Functions

- **int erase**(const std::string &payload, const std::string &datum) const
  
  *Send a request to the MCS to delete data.*

- **int exec**(const std::string &payload, const std::string &command) const
  
  *Send a request to the MCS to execute a command.*

- **int fetch**(const std::string &payload, const std::string &datum) const
  
  *Send a request to the MCS to download data.*

- **int kill**(const std::string &payload, const std::string &experiment) const
  
  *Send a request to the MCS to stop an experiment.*

- **int list**(const std::string &payload, const std::string &path) const
  
  *Send a request to the MCS to get a list of data in a path.*

- **McsClient**(const std::string &**myUrl**, const std::string &serverUrl)
  
  *Constructor of MCS client object.*

- **int run**(const std::string &payload, const std::string &experiment) const
  
  *Send a request to the MCS to start an experiment.*

- **int schedule**(const std::string &payload, const std::string &experiment, const int start) const
  
  *Send a request to the MCS to schedule an experiment.*

- **int store**(const std::string &payload, const ExperimentParts &experiment) const
  
  *Send a request to the MCS.*
CHAPTER 3. COMPONENT DESCRIPTIONS

Protected Attributes

- `xmlrpc_c::carriageParm_curl0 * myCarriageParmPtr`
  The parameters used for the connection.

- `xmlrpc_c::client_xml * myClientPtr`
  The actual XML-RPC client.

- `std::string myUrl`
  The URL that can be used to report.

Constructor & Destructor Documentation

`McsClient (const std::string & myUrl, const std::string & serverUrl)` [inline]

Parameters:

- `myUrl` The local URL the server can use for his response.
- `serverUrl` The URL for the server.

Precondition:

true

Member Function Documentation

`int erase (const std::string & payload, const std::string & datum) const`

Parameters:

- `payload` The payload on which the datum has to be deleted.
- `datum` The datum that has to be deleted.

Precondition:

true

Returns:

The id that will be used when the answer is send.

`int exec (const std::string & payload, const std::string & command) const`

Parameters:

- `payload` The payload on which the command has to be executed.
- `command` The command that has to be executed.

Precondition:

true

Returns:

The id that will be used when the answer is send.
int fetch (const std::string & payload, const std::string & datum) const

Parameters:

payload The payload on which the command has to be executed.
datum The datum that has to be downloaded.

Precondition:
true

Returns:
The id that will be used when the answer is send.

int kill (const std::string & payload, const std::string & experiment) const

Parameters:

payload The payload on which the experiment has to be stopped.
experiment The experiment that has to be stopped.

Precondition:
true

Returns:
The id that will be used when the answer is send.

int list (const std::string & payload, const std::string & path) const

Parameters:

payload The payload for which the list is requested.
path The path for which the list is requested.

Precondition:
true

Returns:
The id that will be used when the answer is send.

int run (const std::string & payload, const std::string & experiment) const

Parameters:

payload The payload on which the experiment has to be executed.
experiment The experiment that has to be executed.

Precondition:
true

Returns:
The id that will be used when the answer is send.
**CHAPTER 3. COMPONENT DESCRIPTIONS**

int schedule (const std::string & payload, const std::string & experiment, const int start) const

Parameters:

- **payload** The payload on which the experiment has to be executed.
- **experiment** The experiment that has to be scheduled.
- **start** The time when the experiment has to start.

Precondition:

true

Returns:

The id that will be used when the answer is send.

int store (const std::string & payload, const ExperimentParts & experiment) const

Parameters:

- **payload** The payload to which the experiment has to be send.
- **experiment** The experiment that has to be uploaded.

Precondition:

true

Returns:

The id that will be used when the answer is send.

**Member Data Documentation**

xmlrpc_c::carriageParm_curl0* myCarriageParmPtr [protected, inherited]

xmlrpc_c::client_xml* myClientPtr [protected, inherited]

std::string myUrl [protected, inherited]

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

- server/mcsClient.h
- server/mcsClient.cpp

**3.2.65 McsController Class Reference**

This class manages all connections with the MCSs.

#include <mcsController.h>

 Inherits XmlRpcController.
Inheritance diagram for McsController:

Public Member Functions

- void callbackAcknowledged (Satellite *sat, const int id)
  
  Callback function used for replies to acknowledged (p. 125).

- void callbackFailed (Satellite *sat, const int id, const std::string &remarks)
  
  Callback function used for replies to failed (p. 233).

- void callbackNewData (const Satellite *sat, int payload, const std::string &id, const std::string &type, const std::vector<unsigned char> &data) const
  
  Callback function used for replies to newData (p. 274).

- void checkAllSatellites ()
  
  Check all satellites and initiate a listener and a client for each satellite if they do not exist.

- void checkForTimeouts ()
  
  Check if there are timeouts for requests to MCSes.

- void checkSatellite (Satellite &sat)
  
  Check if there is already a MCS client and listener for a satellite, and if not create them.

- bool erase (const int bookingId, Satellite &sat, const std::string &payload, const std::string &datum)
  
  Send a request to the MCS to delete data.
CHAPTER 3. COMPONENT DESCRIPTIONS

- **bool exec** (const int bookingId, **Satellite** &sat, const std::string &payload, const std::string &command)

  Send a request to the MCS to execute a command.

- **bool fetch** (const int bookingId, **Satellite** &sat, const std::string &payload, const std::string &datum)

  Send a request to the MCS to download data.

- **std::string getHostname ()**

  Retrieve the local hostname from the config file ’server.icf’.

- **bool kill** (const int bookingId, **Satellite** &sat, const std::string &payload, const std::string &experiment)

  Send a request to the MCS to stop an experiment.

- **bool list** (const int bookingId, **Satellite** &sat, const std::string &payload, const std::string &path)

  Send a request to the MCS to get a list of data in a path.

- **McsController (Scheduler *scheduler)**

- **int removeBooking (Satellite *sat, int id)**

  Removes information of the request with identifier id.

- **bool run** (const int bookingId, **Satellite** &sat, const std::string &payload, const std::string &experiment)

  Send a request to the MCS to start an experiment.

- **bool schedule** (const int bookingId, **Satellite** &sat, const std::string &payload, const std::string &experiment, const int start)

  Send a request to the MCS to schedule an experiment.

- **bool store** (const int bookingId, **Satellite** &sat, const std::string &payload, const **ExperimentParts** &experiment)

  Send a request to the MCS.

- **bool unschedule** (const int bookingId, **Satellite** &sat, const std::string &payload, const std::string &experiment)

  Send a request to the MCS to unschedule an experiment.

Protected Member Functions

- **SocketInfo getNewSocket ()**

  A function to get a socket bound to an unused port.

Protected Attributes

- **std::map< **SatelliteID**, std::map< int, int > > bookingIds**

  A map used for mapping server ids to bookingIds per satellite.
• std::map<SatelliteID, McsInfo *> infoList
  A map for storing all XML RPC clients and listeners.

• int lastUsed
  The port that was last used for a XML RPC listener.

• Scheduler * scheduler
  Scheduler (p. 302) used for callbacks.

• std::map<int, int> timeouts
  A map used for mapping server bookingIds to timeouts per satellite.

Constructor & Destructor Documentation

McsController (Scheduler * scheduler)

Member Function Documentation

void callbackAcknowledged (Satellite * sat, const int id)

Parameters:
  sat The satellite for which the reply has been received.
  id The id of the reply.

Precondition:
  true

Postcondition:
  The reply has been forwarded to the scheduler.

void callbackFailed (Satellite * sat, const int id, const std::string & remarks)

Parameters:
  sat The satellite for which the reply has been received.
  id The id of the reply.
  remarks Remarks about the cause of the failure.

Precondition:
  true

Postcondition:
  The reply has been forwarded to the scheduler.
void callbackNewData (const Satellite * sat, int payload, const std::string & id, const std::string & type, const std::vector<unsigned char> & data) const

Parameters:

sat The satellite for which the reply has been received.
payload the payload for which the reply has been received.
id The id for the received data.
type The type of the received data.
data The received data.

Precondition:

true

Postcondition:

The reply has been forwarded to the scheduler.

void checkAllSatellites ()

Postcondition:

A listener and a client exists for each satellite.

void checkForTimeouts ()

Postcondition:

For each request that has timed out a callback is done to the scheduler.

void checkSatellite (Satellite & sat)

Parameters:

sat The satellite that has to be checked.

Precondition:

true

Postcondition:

A MCS client and listener for the given satellite exist.

bool erase (const int bookingId, Satellite & sat, const std::string & payload, const std::string & datum)

Parameters:

bookingId The id that has to be used for reporting the reply.
CHAPTER 3. COMPONENT DESCRIPTIONS

\textit{sat} The satellite to which the message has to be sent.
\textit{payload} The payload on which the datum has to be deleted.
\textit{datum} The datum that has to be deleted.

\textbf{Precondition:}
\begin{itemize}
  \item true
\end{itemize}

\textbf{Postcondition:}
\begin{itemize}
  \item bookingIds[sat.sid][id] = bookingId.
\end{itemize}

\textbf{Returns:}
\begin{itemize}
  \item Whether the message has been sent successfully.
\end{itemize}

\textbf{bool exec (const int bookingId, Satellite & sat, const std::string & payload, const std::string & command)}

\textbf{Parameters:}
\begin{itemize}
  \item \textit{bookingId} The id that has to be used for reporting the reply.
  \item \textit{sat} The satellite to which the message has to be sent.
  \item \textit{payload} The payload on which the command has to be executed.
  \item \textit{command} The command that has to be executed.
\end{itemize}

\textbf{Precondition:}
\begin{itemize}
  \item true
\end{itemize}

\textbf{Postcondition:}
\begin{itemize}
  \item bookingIds[sat.sid][id] = bookingId.
\end{itemize}

\textbf{Returns:}
\begin{itemize}
  \item Whether the message has been sent successfully.
\end{itemize}

\textbf{bool fetch (const int bookingId, Satellite & sat, const std::string & payload, const std::string & datum)}

\textbf{Parameters:}
\begin{itemize}
  \item \textit{bookingId} The id that has to be used for reporting the reply.
  \item \textit{sat} The satellite to which the message has to be sent.
  \item \textit{payload} The payload on which the command has to be executed.
  \item \textit{datum} The datum that has to be downloaded.
\end{itemize}

\textbf{Precondition:}
\begin{itemize}
  \item true
\end{itemize}

\textbf{Postcondition:}
\begin{itemize}
  \item bookingIds[sat.sid][id] = bookingId.
\end{itemize}

\textbf{Returns:}
\begin{itemize}
  \item Whether the message has been sent successfully.
string getHostname () [inherited]

SocketInfo getNewSocket () [protected, inherited]

Precondition:
true

Returns:
A socket bound to a port and that port.

bool kill (const int bookingId, Satellite & sat, const std::string & payload, const std::string & experiment)

Parameters:
bookingId The id that has to be used for reporting the reply.
sat The satellite to which the message has to be sent.
payload The payload on which the experiment has to be stopped.
experiment The experiment that has to be stopped.

Precondition:
true

Postcondition:
bookingIds[sat.sid][id] = bookingId.

Returns:
Whether the message has been sent successfully.

bool list (const int bookingId, Satellite & sat, const std::string & payload, const std::string & path)

Parameters:
bookingId The id that has to be used for reporting the reply.
sat The satellite to which the message has to be send.
payload The payload for which the list is requested.
path The path for which the list is requested.

Precondition:
true

Postcondition:
bookingIds[sat.sid][id] = bookingId.

Returns:
Whether the message has been sent successfully.
int removeBooking (Satellite * sat, int id)  [inherited]

Parameters:
- sat Satellite (p. 293) to which the request belongs.
- id The identifier of the request.

bool run (const int bookingId, Satellite & sat, const std::string & payload, const std::string & experiment)

Parameters:
- bookingId The id that has to be used for reporting the reply.
- sat The satellite to which the message has to be sent.
- payload The payload on which the experiment has to be executed.
- experiment The experiment that has to be executed.

Precondition:
true

Postcondition:
bookingIds[sat.sid][id] = bookingId.

Returns:
Whether the message has been send successfully.

bool schedule (const int bookingId, Satellite & sat, const std::string & payload, const std::string & experiment, const int start)

Parameters:
- bookingId The id that has to be used for reporting the reply.
- sat The satellite to which the message has to be sent.
- payload The payload on which the experiment has to be executed.
- experiment The experiment that has to be scheduled.
- start The time when the experiment has to start.

Precondition:
true

Postcondition:
bookingIds[sat.sid][id] = bookingId.

Returns:
Whether the message has been sent successfully.
bool store (const int bookingId, Satellite & sat, const std::string & payload, const ExperimentParts & experiment)

Parameters:

bookingId The id that has to be used for reporting the reply.
sat The satellite to which the message has to be sent.
payload The payload to which the experiment has to be sent.
experiment The experiment that has to be uploaded.

Precondition:

true

Postcondition:

bookingIds[sat.sid][id] = bookingId.

Returns:

Whether the message has been sent successfully.

bool unschedule (const int bookingId, Satellite & sat, const std::string & payload, const std::string & experiment)

Parameters:

bookingId The id that has to be used for reporting the reply.
sat The satellite to which the message has to be sent.
payload The payload on which the experiment has to be executed.
experiment The experiment that has to be scheduled.

Precondition:

true

Postcondition:

bookingIds[sat.sid][id] = bookingId.

Returns:

Whether the message has been sent successfully.

Member Data Documentation

std::map<SatelliteID, std::map<int, int>> bookingIds [protected, inherited]

std::map<SatelliteID, McsInfo* > infoList [protected]

int lastUsed [protected, inherited]
CHAPTER 3. COMPONENT DESCRIPTIONS

Scheduler* scheduler [protected, inherited]

std::map<int, int> timeouts [protected, inherited]
The documentation for this class was generated from the following files:

- server/mcsController.h
- server/mcsController.cpp

3.2.66 McsInfo Class Reference

This class contains information for communication with an MCS.

```
#include <mcsController.h>
```

Public Attributes

- McsClient * client
- int port
- pthread_t thread

Member Data Documentation

McsClient* client

int port

pthread_t thread

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

- server/mcsController.h

3.2.67 McsListener Class Reference

This class can be used to receive messages from a MCS.

```
#include <mcsListener.h>
```

Inherits XmlRpcListener.
CHAPTER 3. COMPONENT DESCRIPTIONS

Inheritance diagram for McsListener:

```
McsListener
# controller
# sat
+ McsListener()
+ ~McsListener()
XmlRpcListener
# myAbyssServerPtr
+ run()
+ XmlRpcListener()
```

Public Member Functions

- **McsListener** (const int socket, Satellite &sat, McsController &controller)  
  Constructor of MCS listener object.

- **~McsListener ()**  
  Destructor of MCC listener object.

Protected Attributes

- **McsController * controller**  
  Controller (p. 170) used to report incoming messages.

- **Satellite sat**  
  The satellite for which messages are received.

Private Member Functions

- **void run ()**  
  Start listening for incoming messages.

Private Attributes

- **xmlrpc_c::serverAbyss * myAbyssServerPtr**  
  The XML RPC server.

Constructor & Destructor Documentation

McsListener (const int socket, Satellite & sat, McsController & controller)
 Parameters:
   
   socket  The socket that has to be used for listening.
   sat    The satellite for which the listener listens.
   controller  The controller that has to be called when receiving a message.

 Precondition:
   true

 ~McsListener ()

 Member Function Documentation

 void run ()  [inherited]

 Precondition:
   true

 Postcondition:
   The listeners is listening for incoming messages.

 Member Data Documentation

 McsController* controller  [protected]

 xmlrpc_c::serverAbyss* myAbyssServerPtr  [protected, inherited]

 Satellite sat  [protected]
   The documentation for this class was generated from the following files:
   
   - server/mcsListener.h
   - server/mcsListener.cpp

 3.2.68  newData Class Reference

 Inherits xmlrpc_c::method.
CHAPTER 3. COMPONENT DESCRIPTIONS

Inheritance diagram for newData:

```
method

newData
  - controller
  - sat
  + execute()
  + newData()
```

Public Member Functions

- void `execute` (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
- `newData` (Satellite *tmpSat, McsController *tmpController)

Private Attributes

- McsController * controller
- Satellite * sat

Constructor & Destructor Documentation

```
newData (Satellite * tmpSat, McsController * tmpController) [inline]
```

Member Function Documentation

```
void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value *const retvalP) [inline]
```

Member Data Documentation

```
McsController* controller [private]
```

```
Satellite* sat [private]
```

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

```
server/mcsListener.cpp
```

3.2.69 ParamTypes Union Reference

List of possible types for the parameters of a command.

```
#include <defs.h>
```
CHAPTER 3. COMPONENT DESCRIPTIONS

Public Attributes

- bool b
- double d
- int i

Member Data Documentation

bool b
double d
int i

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

- common/defs.h

3.2.70 Payload Class Reference

< std::exception Administration frontend for managing payload entities.
#include <payload.h>

Public Member Functions

- std::vector<ExperimentID> getExperiments() const throw (PayloadException)
  Retrieves the experiment identifier of the experiments defined on this payload.
- PayloadID getID() const
  Returns the identifier for this payload.
- std::vector<CommandID> getListOfCommands() const throw (PayloadException)
  Retrieves identifier of all commands applicable to the payload.
- std::string getName() throw (PayloadException)
  Returns the name of the payload.
- SatelliteID getSatelliteID() throw (PayloadException)
  Returns the identifier of the satellite to which the payload is linked.
- void instantiate() throw (PayloadException)
  Retrieves data from payload with PayloadID from lower layer.
- bool isRegistered() throw (PayloadException)
  Returns the registration status of this satellite.
- Payload(const PayloadID id) throw (PayloadException)
  Constructor of a new payload object with PayloadID id.
CHAPTER 3. COMPONENT DESCRIPTIONS

- **Payload** (const std::string &*name*)
  
  Constructor of a new payload object, which is not yet in the database.

- **Payload** ()
  
  Empty constructor.

- void **registerPayload** ()
  
  Registers the payload with the system; the payload can now be used within the system.

- void **setName** (const std::string &*name*)
  
  Sets the name of the payload.

- void **setSatelliteID** (SatelliteID *s*)
  
  Links the payload to the satellite with identifier *s*.

- void **setUpdate** ()
  
  Indicate that the data needs to be updated.

- void **store** () throw (PayloadException)
  
  Stores the data of the payload into the database.

- void **unregisterPayload** ()
  
  Unregisters the payload from the system; the payload can now not be used within the system.

- ~Payload ()
  
  Destructor of payload object.

**Private Attributes**

- PayloadID **identifier**
  
  Identifier for this object in the lower layer.

- std::string **name**
  
  Name of the payload.

- bool **needs_storage**
- bool **needs_update**
- bool **registered**
  
  Whether this payload is registered.

- SatelliteID **satelliteID**

**Classes**

- class **PayloadException**
  
  Exception class thrown by **Payload** (p. 276).
CHAPTER 3. COMPONENT DESCRIPTIONS

Constructor & Destructor Documentation

Payload ()

Precondition:
true

Postcondition:
this.getName() = "" \ this.isRegistered() = false \ this.getSatelliteId() = -1 \ this.getID() = -1 \ this.needs_update = false \ this.needs_storage = false

Payload (const std::string & name)

Parameters:
name The name of the payload.

Precondition:
true

Postcondition:
this.getName() = name \ this.isRegistered() = false \ this.getSatelliteId() = -1 \ this.getID() = -1 \ this.needs_update = false \ this.needs_storage = false

Payload (const PayloadID id) throw (PayloadException)

Exceptions:
PayloadException (p. 282)

Parameters:
id A PayloadID identifying the payload in the lower layer.

Precondition:
id >= 0 \ (exists p in connection.Payload ( p.id = id))

Postcondition:
this.getName() = "" \ this.isRegistered() = false \ this.getSatelliteId() = -1 \ this.getID() = -1 \ this.needs_update = true \ this.needs_storage = false

~Payload ()
CHAPTER 3. COMPONENT DESCRIPTIONS

Member Function Documentation

vector<ExperimentID> getExperiments() const throw (PayloadException)

Exceptions:

PayloadException (p. 282)

Precondition:
true

Returns:
{e.getID() | (e in connection.Experiment) \ (e.getPayload() = this)}

PayloadID getID() const

Precondition:
true

Returns:
this.identifier

vector<CommandID> getListOfCommands() const throw (PayloadException)

Exceptions:

PayloadException (p. 282)

Precondition:
true

Returns:
{c.getID() | (c in connection.Command) \ (c.getPayload() = this)}

string getName() throw (PayloadException)

Exceptions:

PayloadException (p. 282)

Precondition:
true

Returns:
this.name

DDD 0.1 279
SatelliteID getSatelliteID () throw (PayloadException)

Exceptions:

PayloadException (p. 282)

Precondition:

true

Returns:

this.satelliteID

void instantiate () throw (PayloadException)

Exceptions:

PayloadException (p. 282)

Precondition:

this.getID() >= 0

Postcondition:

The parameters of the payload are instantiated with information from the lower layer /
this.needs_update = false.

bool isRegistered () throw (PayloadException)

Exceptions:

PayloadException (p. 282)

Precondition:

true

Returns:

this.registered

void registerPayload ()

Precondition:

true

Postcondition:

this.isRegistered() = true /
this.needs_storage = true
void setName (const std::string & name)

Parameters:

   name  A string representing the name of the payload.

Precondition:
    true

Postcondition:
    this.getName() = name \ this.needs_storage = true

void setSatelliteID (SatelliteID s)

Precondition:
    this.isRegistered() = true \ s.isRegistered() = true

Postcondition:
    this.getSatellite() = s \ this.needs_storage = true

void setUpdate ()

Precondition:
    this.identifier > -1

Postcondition:
    this.needs_update = true

void store () throw (PayloadException)

Exceptions:

    PayloadException (p. 282)

Precondition:
    true

Postcondition:
    The data of the payload is stored into the database if the data is changed.

void unregisterPayload ()

Precondition:
    true

Postcondition:
    this.isRegistered() = false \ this.needs_storage = true
CHAPTER 3. COMPONENT DESCRIPTIONS

Member Data Documentation

PayloadID identifier [private]

std::string name [private]

bool needs_storage [private]
Whether the data of this payload needs client-server communication to store its attributes.

bool needs_update [private]
Whether the data of this payload needs client-server communication to update its attributes.

bool registered [private]

SatelliteID satelliteID [private]
Identifier of the satellite to which the payload is linked.

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

- common/payload.h
- common/payload.cpp

3.2.71 Payload::PayloadException Class Reference

Exception class thrown by Payload (p. 276).

#include <payload.h>

Inherits runtime_error.

Inheritance diagram for Payload::PayloadException:

```
  + Payload::PayloadException
  + PayloadException()
  runtime_error
```

Public Member Functions

- PayloadException (const std::string &arg)
CHAPTER 3. COMPONENT DESCRIPTIONS

Constructor & Destructor Documentation

PayloadException (const std::string & arg) [inline]
The documentation for this class was generated from the following file:

- common/payload.h

3.2.72 Queue Class Reference

<typename for SatelliteID.

#include <queue.h>

Public Member Functions

- std::list< QueueItem > clearQueue ()
  Returns the contents of the queue and then clears it.

- void deleteHistoryItem (const ItemID &itemID)
  Delete the history items where the itemID is equal to itemID.

- void deleteQueueItem (QueueItem &item)
  Removes an item from the queue.

- bool empty () const
  Returns whether the queue is empty.

- std::list< QueueItem > getHistory () const
  Returns the contents of the history.

- std::list< QueueItem > getQueue () const
  Returns the contents of the queue.

- int historySize () const
  Returns the size of the history.

- void instantiate ()
  Retrieves data from queue with identifier the tuple (satelliteid, type).

- QueueItem popQueueItem ()
  Returns the first element of the queue.

- void pushQueueItem (QueueItem &item)
  Adds a queue item to the queue.

- Queue (const SatelliteID &satelliteID, const BookingType &type)
  Constructor for a queue. A (satelliteID, type) tuple identifies a queue. Which will be instantiated.

- Queue ()
Empty constructor.

- **int queueSize () const**
  
  Returns the size of the queue.

- **QueueItem retrieveHistoryItem (const ItemID &itemID) const**
  
  Retrieve the history item where the itemID is equal to itemID.

- **~Queue ()**

  Destructor of a queue object.

Public Attributes

- **std::list< QueueItem > queueList**

  List of queue items in queue.

- **bool running**

  Whether the queue is running or not.

Private Attributes

- **SatelliteID satelliteID**

- **BookingType type**

  Type of the queue.

Classes

- **class QueueException**

Detailed Description

< std::exception

Constructor & Destructor Documentation

Queue ()

Precondition:

true

Postcondition:

this.queue.empty() = true \ this.history.empty() = true \ satelliteID = -1.
Queue (const SatelliteID & satelliteID, const BookingType & type)

Parameters:

satelliteID Representing the satellite to which the queue belongs.

type A BookingType representing the type of the queue.

Precondition:

true

Postcondition:

this.queue and this.history are instantiated.

~Queue ()

Precondition:

true

Member Function Documentation

std::list< QueueItem > clearQueue ()

Precondition:

true

Returns:

The contents of the queue.

Postcondition:

The queue is empty.

void deleteHistoryItem (const ItemID & itemID)

Parameters:

itemID The itemID on which will be filtered.

Precondition:

true

Postcondition:

(retrieveHistory(itemID)).empty() (p. 286) = true.
void deleteQueueItem (QueueItem & item)

Precondition:
    true

Postcondition:
    item is not in queue.

bool empty () const

Precondition:
    true

Returns:
    this.queue.empty().

std::list< QueueItem > getHistory () const

Precondition:
    true

Returns:
    The contents of the queue.

std::list< QueueItem > getQueue () const

Precondition:
    true

Returns:
    The contents of the queue.

int historySize () const

Precondition:
    true

Returns:
    size of the history.
void instantiate ()

Precondition:
    this.satelliteID() >= 0.

Postcondition:
The parameters of the data are instantiated with information from the lower layer / \ this.needs_
update = false.

QueueItem popQueueItem ()

Precondition:
    true

Returns:
    this.queue[0] when not(this.queue.empty()), otherwise a queueItem with itemID = -1.

void pushQueueItem (QueueItem & item)

Precondition:
    true

Postcondition:
The item is added to the queue.

int queueSize () const

Precondition:
    true

Returns:
    size of the queue.

QueueItem retrieveHistoryItem (const ItemID & itemID) const

Parameters:
    itemID The itemid on which will be filtered.

Precondition:
    true

Returns:
The item in history where the itemID is equal to itemID.
Member Data Documentation

std::list<QueueItem> queueList

bool running

SatelliteID satelliteID  [private]
Satellite (p. 293) to which the queue belongs to. A (satelliteID, type) tuple identifies a queue.

BookingType type  [private]
The documentation for this class was generated from the following files:

- server/queue.h
- server/queue.cpp

3.2.73 Queue::QueueException Class Reference

#include <queue.h>

Inherits runtime_error.

Inheritance diagram for Queue::QueueException:

```
  runtime_error
  + QueueException()
  Queue::QueueException
  + QueueException()
```

Public Member Functions

- QueueException (const std::string &arg)

Detailed Description

Exception class thrown by Queue (p. 283).

Constructor & Destructor Documentation

QueueException (const std::string & arg)  [inline]
The documentation for this class was generated from the following file:

- server/queue.h
3.2.74 QueueItem Class Reference

Specifies a queue item, as used in queues.

```c++
#include <defs.h>
```

Public Member Functions

- `bool operator==(const QueueItem &other) const`
  
  Needed for testing.

Public Attributes

- `Action action`
  
  Action to be performed

- `bool history`
  
  Bool to set if the queue item is in history or not

- `int intVal`
  
  To store an integer value in a queue item.

- `ItemID itemID`
  
  Identifies the queue item.

- `PayloadID payloadID`
  
  Identifies the payload to which the queue item belongs to.

- `SatelliteID satID`
  
  Identifies the satellite to which the item belongs to.

- `std::string strVal`
  
  To store a string value in a queue item.

- `BookingType type`
  
  Type of the queue item.

Member Function Documentation

```c++
bool operator==(const QueueItem & other) const [inline]
```

Member Data Documentation

Action action

Action to be performed

```c++
bool history
```

Bool to set if the queue item is in history or not

```c++
int intVal
```
CHAPTER 3. COMPONENT DESCRIPTIONS

ItemID itemID

PayloadID payloadID

SatelliteID satID

std::string strVal

BookingType type

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

- common/defs.h

3.2.75 Queues Class Reference

Class that contains an internal and an external queue.

#include <scheduler.h>

Public Member Functions

- Queues ()

Public Attributes

- Queue * external
  
  *External queue.*

- Queue * internal
  
  *Internal queue.*

Constructor & Destructor Documentation

Queues () [inline]

Member Data Documentation

Queue* external

Queue* internal

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

- server/scheduler.h
3.2.76 requestFailure Class Reference

Inherits `xmlrpc_c::method`.

Inheritance diagram for requestFailure:

```
method

requestFailure
- controller
- sat
+ execute()
+ requestFailure()
method
```

Public Member Functions

- `void execute (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)`
- `requestFailure (Satellite *tmpSat, MccController *tmpController)`

Private Attributes

- `MccController *controller`
- `Satellite *sat`

Constructor & Destructor Documentation

`requestFailure (Satellite * tmpSat, MccController * tmpController)` [inline]

Member Function Documentation

`void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value *const retvalP)` [inline]

Member Data Documentation

`MccController* controller` [private]

`Satellite* sat` [private]

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

- `server/mccListener.cpp`
3.2.77 requestRejected Class Reference

Inherits xmlrpc_c::method.

Inheritance diagram for requestRejected:

```
method

requestRejected
- controller
- sat
+ execute()
+ requestRejected()
```

Public Member Functions

- void `execute(xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)`
- `requestRejected(Satellite *tmpSat, MccController *tmpController)`

Private Attributes

- `MccController * controller`
- `Satellite * sat`

Constructor & Destructor Documentation

`requestRejected(Satellite * tmpSat, MccController * tmpController) [inline]`

Member Function Documentation

`void execute(xmlrpc_c::paramList const & paramList, xmlrpc_c::value * const retvalP) [inline]`

Member Data Documentation

`MccController * controller [private]`

`Satellite * sat [private]`

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

`server/mccListener.cpp`
3.2.78 Satellite Class Reference

< std::exception Administration frontend for managing satellite entities.

#include <satellite.h>

Public Member Functions

- **ExperimentID getExperiment** (unsigned int time) const throw (SatelliteException)
  
  Returns the experiment that is running at the satellite at a given time, if no experiment is running, return -1.

- **SatelliteID getID** () const
  
  Returns the identifier for this satellite.

- **std::string getMccUrl** () throw (SatelliteException)
  
  Returns the URL at which the MCC of this satellite can be contacted.

- **std::string getMcsUrl** () throw (SatelliteException)
  
  Returns the URL at which the MCS of this satellite can be contacted.

- **std::string getName** () throw (SatelliteException)
  
  Returns the name of this satellite.

- **void instantiate** () throw (SatelliteException)
  
  Retrieves data from satellite with identifier from lower layer.

- **bool isRegistered** () throw (SatelliteException)
  
  Returns the registration status of this satellite.

- **void registerSatellite** ()
  
  Registers this satellite to be used.

- **Satellite** (const SatelliteID id) throw (SatelliteException)
  
  Constructor of a new satellite object with SatelliteID id.

- **Satellite** (const std::string &mccURL, const std::string &mcsUrl, const std::string &name)
  
  Constructor of a new satellite object, which is not yet in the database.

- **Satellite** ()
  
  Empty constructor.

- **void setMccUrl** (const std::string &mccURL)
  
  Sets the URL at which the MCC of this satellite can be contacted.

- **void setMcsUrl** (const std::string &mcsUrl)
  
  Sets the URL at which the MCS of this satellite can be contacted.

- **void setName** (const std::string &newName)
CHAPTER 3. COMPONENT DESCRIPTIONS

Sets the name of this satellite.

- void `setUpdate()`
  Indicate that the data needs to be updated.

- void `store()` throw (SatelliteException)
  Stores the data of the satellite into the database.

- void `unregisterSatellite()`
  Unregisters this satellite, such that it can no longer be used.

- `~Satellite()`
  Destructor of satellite object.

Private Attributes

- `SatelliteID identifier`
  Identifier for this object in the lower layer.

- `std::string mccUrl`
  Location of the MCC for this satellite.

- `std::string mcsUrl`
  Location of the MCS for this satellite.

- `std::string name`
  Name of this satellite.

- bool `needs_storage`
- bool `needs_update`
- bool `registered`
  Whether this satellite is registered.

Classes

- class `SatelliteException`
  Exception class thrown by `Satellite` (p. 293).

Constructor & Destructor Documentation

Satellite()

Precondition:

```
true
```

Postcondition:

```
this.identifier = -1 / this.mccURL = "" / this.mcsURL = "" / this.registered = false / this.name = "" / this.needs_update = false / this.needs_storage = false
```
CHAPTER 3. COMPONENT DESCRIPTIONS

Satellite (const std::string & mccURL, const std::string & mcsUrl, const std::string & name)

Parameters:

- **mccURL** The URL at which the MCC can be contacted.
- **mcsUrl** The URL at which the MCS can be contacted.
- **name** The name of the satellite.

Precondition:

true

Postcondition:

- this.getMccUrl() = mccUrl
- this.getMcsUrl() = mcsUrl
- this.identifier = -1
- this.registered = false
- this.name = name
- this.needs_update = false
- this.needs_storage = false

Satellite (const SatelliteID id) throw (SatelliteException)

Exceptions:

- **SatelliteException** (p. 299)

Parameters:

- **id** a SatelliteID identifying the satellite in the lower layer.

Precondition:

id >= 0

Postcondition:

- this.identifier = id
- this.mccURL = ""
- this.mcsUrl = ""
- this.registered = false
- this.name = ""
- this.needs_update = true
- this.needs_storage = false

~Satellite ()

Precondition:

true

Member Function Documentation

ExperimentID getExperiment (unsigned int time) const throw (SatelliteException)

Exceptions:

- **SatelliteException** (p. 299)

Parameters:

- **time** A time.
Precondition:
true

Returns:
the experiment which is running at time time, else -1.

SatelliteID getID () const

Precondition:
true

Returns:
identifier

std::string getMccUrl () throw (SatelliteException)

Precondition:
true

Returns:
this.mccURL

std::string getMcsUrl () throw (SatelliteException)

Precondition:
true

Returns:
this.mcsUrl

std::string getName () throw (SatelliteException)

Precondition:
true

Returns:
this.name
void instantiate () throw (SatelliteException)

Exceptions:

\textit{SatelliteException} (p. 299)

Precondition:

\texttt{identifier} >= 0

Postcondition:

The parameters of satellites are instantiated with information from the lower layer / \ this.needs._update = false

bool isRegistered () throw (SatelliteException)

Precondition:

true

Returns:

registered

void registerSatellite ()

Precondition:

true

Postcondition:

this.isRegistered() = true / \ this.needs_storage = true

void setMccUrl (const std::string & \texttt{mccURL})

Parameters:

\texttt{mccURL} a string representing the url for the MCC.

Precondition:

true

Postcondition:

this.getMccUrl() = \texttt{mccURL} / \ this.needs_storage = true
void setMcsUrl (const std::string & mcsUrl)

Parameters:

mcsUrl a string representing the url for the MCS.

Precondition:

ture

Postcondition:

this.getMcsUrl() = mcsUrl / this.needs_storage = true

void setName (const std::string & newName)

Parameters:

newName The new name of the satellite.

Precondition:

ture

Postcondition:

this.name = newName / this.needs_storage = true

void setUpdate ()

Precondition:

this.identifier > -1

Postcondition:

this.needs_update = true

void store () throw (SatelliteException)

Exceptions:

SatelliteException (p. 299)

Precondition:

ture

Postcondition:

The data of the satellite is stored into the database if the data is changed.
void unregisterSatellite ()

Precondition:
    true

Postcondition:
    this.isRegistered() = false /\ this.needs_storage = true

Member Data Documentation

SatelliteID identifier [private]

std::string mccUrl [private]

std::string mcsUrl [private]

std::string name [private]

bool needs_storage [private]
    Whether the data of this satellite needs client-server communication to store its attributes.

bool needs_update [private]
    Whether the data of this satellite needs client-server communication to update its attributes.

bool registered [private]

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

- common/satellite.h
- common/satellite.cpp

3.2.79 Satellite::SatelliteException Class Reference

Exception class thrown by Satellite (p. 293).

#include <satellite.h>

Inherits runtime_error.
Inheritance diagram for Satellite::SatelliteException:

```
Satellite::SatelliteException
  + SatelliteException()
  runtime_error
```

Public Member Functions

- **SatelliteException** (const std::string &arg)

**Constructor & Destructor Documentation**

**SatelliteException (const std::string & arg)** [inline]

The documentation for this class was generated from the following file:
- common/satellite.h

3.2.80 SchedInfo Class Reference

A class that contains information needed by the thread that checks a satellite.

```
#include <scheduler.h>
```

**Public Attributes**

- Satellite sat
- Scheduler* scheduler
- BookingType type

**Member Data Documentation**

Satellite sat

Scheduler* scheduler

BookingType type

The documentation for this class was generated from the following file:
- server/scheduler.h
3.2.81 scheduleExperiment Class Reference

Requests to schedule an experiment.

#include <clientconnectivity.h>

Inherits xmlrpc_c::method.

Inheritance diagram for scheduleExperiment:

Public Member Functions

- void execute (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvalP)
- scheduleExperiment (Controller *tmpController)

Private Attributes

- Controller * controller

Detailed Description

Parameters:

- **sessionId** The first parameter, identifying the requesting session.
- **experimentID** The second parameter, an ExperimentID identifying the experiment to be scheduled.
- **startTime** The third parameter, an integer timestamp at which the experiment is to begin.
- **endTime** The fourth parameter, an integer timestamp at which the experiment is to end.
- **gssid** The fifth parameter, a string identifying the ground station at which the booking is to be made.

Precondition:

exists e in Experiment (p. 221): e.getID() = experimentID

Postcondition:

forall e in Experiment (p. 221): e.getID() = experimentID: e.getSchedule = (startTime, endTime)

Constructor & Destructor Documentation

scheduleExperiment (Controller * tmpController)
Member Function Documentation

```cpp
void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value ∗ const retvalP)
```

Member Data Documentation

```cpp
Controller* controller [private]
```

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

- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.82 Scheduler Class Reference

```cpp
#include <scheduler.h>
```

Public Member Functions

- void `callbackBooking` (Satellite &s, const BookingID id, const BookingStatus bookingStatus, const BookingInfo &bookingInfo)
  
  *Pass on the received status j for request i from satellite s.*

- void `callbackListOfOpportunities` (Satellite &s, const BookingID id, const bool status, const BookingList &listOfBookingInfos)
  
  *Pass on the list of opportunities, which were requested.*

- void `callbackMCS` (const Satellite &s, const ItemID id, const bool status, const std::string &remarks)
  
  *Receive the status of a sent request.*

- void `callbackNewData` (const Satellite &s, int payload, const std::string &type, const std::vector<unsigned char> &data)
  
  *Receives new data d for the satellite with identifier s.*

- void `callbackTimeoutMCC` (BookingID id)
  
  *Receive a time out on a certain booking.*

- void `callbackTimeoutMCS` (ItemID id)
  
  *Receive a time out on a certain item.*

- void `checkAllSatellites` ()
  
  *Check if there is already an internal and external queue for all satellites, and if not create them.*

- void `checkSatellite` (SatelliteID sid)
  
  *Check if there is already an internal and external queue for a satellite, and if not create them.*

- void `createBooking` (ExperimentID id, BookingInfo bookingInfo)
create an external booking for an experiment. Note that this does not request a booking to the MCC, but only sets it. The actual requesting will be done when uploadexperiment is called.

- void deleteData (DataID id)
  Deletes the data with identifier d from the satellite from which it was retrieved.

- void downloadHighResolutionData (DataID id)
  Downloads the high-resolution data of the thumbnail with data identifier d.

- void executeCommand (Satellite s, ExperimentID id, std::string c)
  Executes the command c on payload p on satellite s.

- std::vector<BookingID> getBookings (ExperimentID id)
  Get the realtime bookings for an experiment.

- int getListOfOpportunities (const ExperimentID id, const Interval &period)
  User request a list of opportunities in the given interval.

- bool isWindowOfOpportunity (PayloadID id, int ti)
  Returns whether there is a window of opportunity for the payload with payload identifier pld on time ti.

- void processSatellite (Satellite &sat, BookingType type)
  Process the queues of the given satellite.

- void run ()
  Start the thread that periodically tries to send all data in the queues.

- Scheduler (Controller *controller)
  Constructor of the scheduler thread.

- Scheduler ()
  Default constructor.

- void unscheduleExperiment (ExperimentID id)
  Unschedules the experiment.

- bool uploadExperiment (SatelliteID satelliteid, PayloadID payloadid, ExperimentID experimentid)
  Uploads experiment e. Experiment (p. 221) e is added to the internal upload queue and all external bookings associated to the experiment are requested to the MCC.

- ~Scheduler ()
  Default destructor.

Private Member Functions

- bool AppearsIn (std::vector<ExperimentID> list, ExperimentID id)
  Checks if a list of experimentids contains a certain experimentid.
• void cancelExternalBookingsOfExperiment (ExperimentID id)
  Cancels the external bookings of a certain experiment.

• ExperimentParts experimentListToParts (ListOfExperimentCommands list)
  Convert a ListOfExperimentCommands used by the experiment class to ExperimentParts used by the MCS class.

• std::vector< SatelliteID > getAvailableSatellites (int time)
  Get all satellites that have a pass at the given time.

• MccController * getMccController ()
  Get the MCC controller with the right type.

• McsController * getMcsController ()
  Get the MCS controller with the right type.

• ExperimentID getNearestScheduledFrom (std::list< QueueItem > itemList)
  Get experiment which has the nearest scheduledFrom time in the list.

• std::vector< ExperimentID > getUniqueListOfExperiment (std::list< QueueItem > queuelist)
  returns a list where experimentids are unique.

• bool processItem (QueueItem &item)
  Process a certain item to be send to the MCS.

• std::vector< QueueItem > processQueue (Queue *queue)
  Process the given queue.

Private Attributes

• Controller * controller
  The controller that has to be used for callbacks.

• MccController * mccController
  The MCC controller that has to be called for communication with an MCC.

• McsController * mcsController
  The MCS controller that has to be called for communication with an MCS.

• std::map< SatelliteID, Queues > queueList
  A list with internal and external queues per satellite.

• std::vector< pthread_t > threadList
  A list of threads that have to be destroyed when the scheduler is destroyed.
Classes

- class SchedulerException
  
  Exception thrown by the scheduler.

Constructor & Destructor Documentation

Scheduler ()

Precondition:

  true

Scheduler (Controller ∗ controller)

Precondition:

  true

~Scheduler ()

Member Function Documentation

bool AppearsIn (std::vector< ExperimentID > list, ExperimentID id) [private]

Parameters:

  list A list of experimentids.
  id A certain experimentid.

Precondition:

  true

Returns:

  whether id appears in the list.

void callbackBooking (Satellite & s, const BookingID id, const BookingStatus bookingStatus, const BookingInfo & bookingInfo)

Parameters:

  s A Satellite (p. 293) object.
  id A BookingID.
  bookingStatus The status of the booking with BookingID id.
  bookingInfo Info about the booking.
Precondition:
   true

Postcondition:
The status s and the info i is passed on.

Todo
Here the controller could be informed that these experiments could not be uploaded in time
by means of a callback. The controller will then be able to perform some actions such as
informing the client.

`void callbackListOfOpportunities (Satellite & s, const BookingID id, const bool status, const BookingList & listOfBookingInfos)`

Parameters:
   s A Satellite (p. 293) object.
   id A booking id.
   `status` Whether there was no error retrieving the list.
   `listOfBookingInfos` A list of available passes.

Todo
Here the controller could be informed that these experiments could not be uploaded in time
by means of a callback. The controller will then be able to perform some actions such as
informing the client.

Todo
Here the controller could be informed that these experiments could not be uploaded in time
by means of a callback. The controller will then be able to perform some actions such as
informing the client.

`void callbackMCS (const Satellite & s, const ItemID id, const bool status, const std::string & remarks)`

Parameters:
   s A Satellite (p. 293) object.
   id A booking id.
   `status` Whether there was no error sending the request.
   `remarks` Possible remarks about the reason of the failure.

`void callbackNewData (const Satellite & s, int payload, const std::string id, const std::string type, const std::vector<unsigned char> & data)`

Parameters:
   s A Satellite (p. 293) object.
**CHAPTER 3. COMPONENT DESCRIPTIONS**

.payload The payload from which the data is received.

.id The dataID for the received data.

.type The type of the received data.

.data A Data (p. 199) object.

**Precondition:**

(Exists a s’ in Satellites :: s’ = s).

**Postcondition:**

The scheduler updates its queues and sends data d to the controller.

**void callbackTimeoutMCC (BookingID id)**

**Parameters:**

.id The bookingid concerned.

**Precondition:**

true

**Postcondition:**

Not implemented.

**Todo**

Implement: MCC informs scheduler of a Time out on a certain booking after which the controller can try to request to booking again.

**void callbackTimeoutMCS (ItemID id)**

**Parameters:**

.id The itemid concerned.

**Precondition:**

true

**Postcondition:**

Not implemented.

**Todo**

Implement: MCS informs scheduler of a Time out on a certain booking after which the controller can try to request to booking again.
void cancelExternalBookingsOfExperiment (ExperimentID id) [private]

Parameters:

id  the experiment id of which external bookings are to be cancelled (p. 158).

Precondition:

true

Postcondition:

The cancellation of external bookings of this experiment have been requested to the MCC.

void checkAllSatellites ()

Precondition:

true

Postcondition:

A internal and external queue for all satellites exist.

void checkSatellite (SatelliteID sid)

Parameters:

sid  The satellite ID of the satellite that has to be checked.

Precondition:

true

Postcondition:

A internal and external queue for the given satellite exist.

void createBooking (ExperimentID id, BookingInfo bookingInfo)

Parameters:

id  The experiment id.

bookingInfo  Various info about the booking.

Precondition:

true

Postcondition:

The booking has been made in the booking class w.r.t. the experiment.
void deleteData (DataID id)

Parameters:

id a DataID.

Precondition:

There exists a data element in the database with data identifier d.

Postcondition:

The delete command w.r.t the data with identifier d is added to the internal upload queue.

void downloadHighResolutionData (DataID id)

Parameters:

id a DataID.

Precondition:

Identifier d is the data identifier of a thumbnail that exists in the database.

Postcondition:

The high resolution data of the thumbnail with data identifier d is added to the internal download queue.

Todo

Implementation of downloading high resolution data is not done.

void executeCommand (Satellite s, ExperimentID id, std::string c)

Parameters:

s A Satellite (p. 293) object.

id An ExperimentID for which the command is meant.

c a command.

Precondition:

true

Postcondition:

This command has been placed in the queue.

ExperimentParts experimentListToParts (ListOfExperimentCommands list) [private]

Parameters:

list The ListOfExperimentCommands that has to be converted.

Returns:

The converted ExperimentParts.
std::vector<SatelliteID> getAvailableSatellites (int time) [private]

Parameters:
  time The time for which the satellites have to be selected.

Returns:
  The ID’s of the satellites that have a pass at the given time.

std::vector<BookingID> getBookings (ExperimentID id)

Parameters:
  id The experimentid.

Precondition:
  (Exists e’ in Experiment (p.221): e’.id = id).

Returns:
  Associated bookings.

int getListOfOpportunities (const ExperimentID id, const Interval & period)

Parameters:
  id The experimentid for which the list of opportunities is requested.  
  period A interval over which a booking is required.

Returns:
  a unique id identifying the request, -1 if the request has not been send to the MCC.

MccController * getMccController () [private]

McsController * getMcsController () [private]

ExperimentID getNearestScheduledFrom (std::list<QueueItem> itemList) [private]

Parameters:
  itemList a list of queueitems.

Precondition:
  At least 1 queueItem in the list.

Returns:
  The experiment with the nearest scheduledFrom time.
### CHAPTER 3. COMPONENT DESCRIPTIONS

std::vector< ExperimentID > getUniqueListOfExperiment (std::list< QueueItem > queuelist) [private]

**Parameters:**
- queuelist contains experimentids, possible multiple same values.

**Precondition:**
true

**Returns:**
a list of unique experimentids.

bool isWindowOfOpportunity (PayloadID id, int ti)

**Parameters:**
id A PayloadID.
ti A time.

**Precondition:**
(exists p in Payloads :: p.id = pId).

**Postcondition:**
(exists b in Booking (p. 150) :: b.Start <= ti <= b.End).

**Todo**
Implementation of this function has not been done. It could be used by the scheduler to find out whether there is a window of opportunity.

bool processItem (QueueItem & item) [private]

**Parameters:**
item a certain item in the queue.

**Precondition:**
true

**Returns:**
Whether the item has been sent to the MCS.

std::vector< QueueItem > processQueue (Queue * queue) [private]

**Parameters:**
queue The queue that has to be processed.
Postcondition:
All items in the queue are delivered to the MCS.

Returns:
All items that could not be delivered.

Todo
Add time constraint, so if something goes wrong the thread will stop eventually.

Repeat in case of failure / put back in the queue after 3 times

void processSatellite (Satellite & sat, BookingType type)

Parameters:
sat The satellite for which the queues have to be processed.
type The type of the booking.

Todo
Here the controller could be informed that these items have not been processed by means of a callback. The controller will then be able to perform some actions such as informing the client.

void run ()

void unscheduleExperiment (ExperimentID id)

Parameters:
id The experiment which is to be unscheduled.

Precondition:
true

Postcondition:
Experiment (p. 221) with identifier id is unscheduled.

bool uploadExperiment (SatelliteID satelliteid, PayloadID payloadid, ExperimentID experimentid)

Parameters:
satelliteid The SatelliteID of the experiment.
payloadid The PayloadID of the experiment.
experimentid The ExperimentID of the experiment.

Precondition:
There is an e in Experiments :: e.id = eId.

Returns:
If the upload experiment has been performed succesfully.
CHAPTER 3. COMPONENT DESCRIPTIONS

Member Data Documentation

Controller* controller [private]

MccController* mccController [private]

McsController* mcsController [private]

std::map<SatelliteID, Queues> queueList [private]

std::vector<pthread_t> threadList [private]

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

- server/scheduler.h
- server/scheduler.cpp

3.2.83 Scheduler::SchedulerException Class Reference

Exception thrown by the scheduler.

#include <scheduler.h>

Inherits runtime_error.

Inheritance diagram for Scheduler::SchedulerException:

```
    runtime_error
     |
     v
Scheduler::SchedulerException
     |
     v
* SchedulerException()
```

Public Member Functions

- SchedulerException (const std::string &arg)

Constructor & Destructor Documentation

SchedulerException (const std::string & arg) [inline]

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

- server/scheduler.h
3.2.84 sha2 Class Reference

#include <sha2.h>

Public Types

- enum SHA_TYPE {
  enuSHA_NONE, enuSHA1, enuSHA160 = enuSHA1, enuSHA224,
  enuSHA256, enuSHA384, enuSHA512, enuSHA_LAST }

Public Member Functions

- void End ()
- SHA_TYPE GetEnumType ()
- const std::string & GetHash (SHA_TYPE type, const sha_byte *data, size_t len)
- const char * GetTypeString ()
- const char * HexHash ()
- void Init (SHA_TYPE type)
- bool IsBigEndian ()
- const char * RawHash (int &length)
- sha2 ()
- const std::string & StringHash ()
- void Update (const sha_byte *data, size_t len)

Private Member Functions

- void ADDINC128 (sha_word64 *w, sha_word32 n)
- sha_word64 Ch (sha_word64 x, sha_word64 y, sha_word64 z)
- sha_word32 Ch (sha_word32 x, sha_word32 y, sha_word32 z)
- sha_word64 Maj (sha_word64 x, sha_word64 y, sha_word64 z)
- sha_word32 Maj (sha_word32 x, sha_word32 y, sha_word32 z)
- void MEMCPY_BCOPY (void *d, const void *s, size_t l)
- void MEMSET_BZERO (void *p, size_t l)
- sha_word32 Parity (sha_word32 x, sha_word32 y, sha_word32 z)
- void REVERSE32 (sha_word32 w, sha_word32 &x)
- void REVERSE64 (sha_word64 w, sha_word64 &x)
- sha_word32 ROTL32 (sha_word32 b, sha_word32 x)
- sha_word32 ROTR32 (sha_word32 b, sha_word32 x)
- sha_word64 ROTR64 (sha_word64 b, sha_word64 x)
- void SHA1_Internal_Transform (const sha_word32 *data)
- void SHA256_Internal_Last (bool isSha1=false)
- void SHA256_Internal_Transform (const sha_word32 *data)
  * SHA-256: *****************************************************
- void SHA32bit_Update (const sha_byte *data, size_t len, bool isSha1=false)
- void SHA512_Internal_Last ()
- void SHA512_Internal_Transform (const sha_word64 *)
- void SHA64bit_Update (const sha_byte *data, size_t len)
- sha_word64 SHR (sha_word64 b, sha_word64 x)
CHAPTER 3. COMPONENT DESCRIPTIONS

- sha_word32 SHR (sha_word32 b, sha_word32 x)
- sha_word32 sigma0_256 (sha_word32 x)
- sha_word32 Sigma0_256 (sha_word32 x)
- sha_word64 sigma0_512 (sha_word64 x)
- sha_word64 Sigma0_512 (sha_word64 x)
- sha_word32 sigma1_256 (sha_word32 x)
- sha_word32 Sigma1_256 (sha_word32 x)
- sha_word64 sigma1_512 (sha_word64 x)
- sha_word64 Sigma1_512 (sha_word64 x)

Private Attributes

- sha2::SHA_CTX ctx
- bool m_boolEnded
- bool m_boolIsBigEndian
- char m_chrHexHash [SHA512_DIGESTC_STRING_LENGTH]
- char m_chrRawHash [SHA512_DIGESTC_LENGTH]
- sha_byte m_digest [SHA512_DIGESTC_LENGTH]
- std::string m_strHash
- SHA_TYPE m_Type

Classes

- struct SHA_CTX

Member Enumeration Documentation

def SHA_TYPE

Enumerator:

- enumSHA_NONE
- enumSHA1
- enumSHA160
- enumSHA224
- enumSHA256
- enumSHA384
- enumSHA512
- enumSHA_LAST

Constructor & Destructor Documentation

sha2 () [inline]

Member Function Documentation

void ADDINC128 (sha_word64 * w, sha_word32 n) [inline, private]
CHAPTER 3. COMPONENT DESCRIPTIONS

sha_word64 Ch (sha_word64 x, sha_word64 y, sha_word64 z) [inline, private]

sha_word32 Ch (sha_word32 x, sha_word32 y, sha_word32 z) [inline, private]

void End ()

SHA_TYPE GetEnumType () [inline]

const string & GetHash (SHA_TYPE type, const byte * data, size_t len)

const char* GetTypeString () [inline]

const char * HexHash ()

void Init (SHA_TYPE type)

bool IsBigEndian () [inline]

sha_word64 Maj (sha_word64 x, sha_word64 y, sha_word64 z) [inline, private]

sha_word32 Maj (sha_word32 x, sha_word32 y, sha_word32 z) [inline, private]

void MEMCPY_BCOPY (void * d, const void * s, size_t l) [inline, private]

void MEMSET_BZERO (void * p, size_t l) [inline, private]

sha_word32 Parity (sha_word32 x, sha_word32 y, sha_word32 z) [inline, private]

const char * RawHash (int & length)

void REVERSE32 (sha_word32 w, sha_word32 & x) [inline, private]

void REVERSE64 (sha_word64 w, sha_word64 & x) [inline, private]

sha_word32 ROTL32 (sha_word32 b, sha_word32 x) [inline, private]

sha_word32 ROTR32 (sha_word32 b, sha_word32 x) [inline, private]

sha_word64 ROTR64 (sha_word64 b, sha_word64 x) [inline, private]

void SHA1_Internal_Transform (const sha_word32 * data) [private]
CHAPTER 3. COMPONENT DESCRIPTIONS

void SHA256_Internal_Last (bool isSha1 = false) [private]

void SHA256_Internal_Transform (const sha_word32 * data) [private]

void SHA32bit_Update (const sha_byte * data, size_t len, bool isSha1 = false) [private]

void SHA512_Internal_Last () [private]

void SHA512_Internal_Transform (const sha_word64 *) [private]

void SHA64bit_Update (const sha_byte * data, size_t len) [private]

sha_word64 SHR (sha_word64 b, sha_word64 x) [inline, private]

sha_word32 SHR (sha_word32 b, sha_word32 x) [inline, private]

sha_word32 sigma0_256 (sha_word32 x) [inline, private]

sha_word32 Sigma0_256 (sha_word32 x) [inline, private]

sha_word64 sigma0_512 (sha_word64 x) [inline, private]

sha_word64 Sigma0_512 (sha_word64 x) [inline, private]

sha_word32 sigma1_256 (sha_word32 x) [inline, private]

sha_word32 Sigma1_256 (sha_word32 x) [inline, private]

sha_word64 sigma1_512 (sha_word64 x) [inline, private]

sha_word64 Sigma1_512 (sha_word64 x) [inline, private]

const string & StringHash ()

void Update (const sha_byte * data, size_t len)

Member Data Documentation

struct sha2::SHA_CTX ctx [private]

bool m_boolEnded [private]
bool m_bIsBigEndian [private]

char m_chrHexHash[SHA512_DIGESTC_STRING_LENGTH] [private]  
char m_chrRawHash[SHA512_DIGESTC_LENGTH] [private]
sha_byte m_digest[SHA512_DIGESTC_LENGTH] [private]
std::string m_strHash [private]
SHA_TYPE m_Type [private]

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

- common/sha2.h
- common/sha2.cpp

### 3.2.85 sha2::SHA_CTX Struct Reference

**Public Attributes**

- sha_word64 bitcount [2]
- sha_byte buffer [128]
- sha_byte state [sizeof(sha_word64)*8]

**Member Data Documentation**

sha_word64 bitcount[2]

sha_byte buffer[128]

sha_byte state[8]

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

- common/sha2.h

### 3.2.86 SingleLog Class Reference

Specifies a log item.

```
#include <defs.h>
```

**Public Types**

- enum Component { MCS, MCC, Controller, Scheduler }

  Enumeration of components to which the log item belongs to.
CHAPTER 3. COMPONENT DESCRIPTIONS

Public Attributes

- **ExperimentID experimentID**
  
  *Experiment (p. 221) the log concerns (can be empty).*

- **std::string message**
  
  *Message in log.*

- **PayloadID payloadID**
  
  *Payload (p. 276) the log concerns (can be empty).*

- **int priority**
  
  *Message has a certain priority.*

- **SatelliteID satelliteID**

- **Component source**
  
  *Component responsible for creating log.*

- **int time**
  
  *Date/Time when this object was created.*

Member Enumeration Documentation

```cpp
enum Component
{
    MCS,
    MCC,
    Controller,
    Scheduler
}
```

Member Data Documentation

- **ExperimentID experimentID**
  
  *std::string message*

- **PayloadID payloadID**

- **int priority**
  
  *Satellite (p. 293) the log concerns (can be empty)*

- **Component source**
int time

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

- common/defs.h

### 3.2.87 SocketInfo Struct Reference

Convenience struct.

```c
#include <xmlRpcController.h>
```

#### Public Attributes

- int `port`
- int `socket`

Member Data Documentation

```c
int port
```

```c
int socket
```

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

- server/xmlRpcController.h

### 3.2.88 runtime_error Class Reference

Inherited by `Account::AccountException`, `AdminLowlevel::AdminLowlevelException`, `Booking::BookingException`, `Command::CommandException`, `Controller::ControllerException`, `Data::DataException`, `Experiment::ExperimentException`, `LogException`, `Payload::PayloadException`, `Queue::QueueException`, `Satellite::SatelliteException`, `Scheduler::SchedulerException`, and `XmlRpcController::XmlRpcControllerException`.

Inheritance diagram for runtime_error:

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

- server/booking.h
- server/controller.h
- common/controller.h
- common/experiment.h
3.2.89 unscheduleExperiment Class Reference

Requests to unschedule an experiment.

#include <clientconnectivity.h>

Inherits xmlrpc_c::method.

Inheritance diagram for unscheduleExperiment:

Public Member Functions

- void execute (xmlrpc_c::paramList const &paramList, xmlrpc_c::value *const retvP)
- unscheduleExperiment (Controller *controller)

Private Attributes

- Controller * controller

Detailed Description

Parameters:

- session The session requesting the unschedule.
- experimentID The id of the Experiment (p. 221) to be unscheduled.

Precondition:

- experimentID >= 0

Postcondition:

The server is requested to unschedule the experiment with experimentID
CHAPTER 3. COMPONENT DESCRIPTIONS

Constructor & Destructor Documentation

unscheduleExperiment (Controller ∗ controller)

Member Function Documentation

void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value ∗ const retvalP)

Member Data Documentation

Controller ∗ controller [private]
The documentation for this class was generated from the following files:

- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.90 updateExperiment Class Reference

Updates the experiment on the server.

#include <clientconnectivity.h>

Inherits xmlrpc_c::method.

Inheritance diagram for updateExperiment:

Public Member Functions

- void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value ∗ const retvalP)
- updateExperiment (Controller ∗ tmpController)

Private Attributes

- Controller ∗ controller

Detailed Description

Parameters:

- sessionID The first parameter, identifying the requesting session.
CHAPTER 3. COMPONENT DESCRIPTIONS

experimentData  An xmlRpc struct containing the information with which the experiment is to be updated.

Precondition:

exists e in Experiment (p. 221): e.id = experimentData.id *

Postcondition:

forall e in Experiment (p. 221): e.id = experimentData.id: e.listOfCommands = experimentData.listOfCommands /
\  e.name = experimentData.name /
\  e.payload = experimentData.payload /
\  

Constructor & Destructor Documentation

updateExperiment (Controller * tmpController)

Member Function Documentation

void execute (xmlrpc_c::paramList const & paramList, xmlrpc_c::value *const retvalP)

Member Data Documentation

Controller* controller  [private]

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

- server/clientconnectivity.h
- server/clientconnectivity.cpp

3.2.91 xmlrpc_c::method Class Reference

Inherited by acknowledged, booked, cancelled, changePassword, createExperiment, failed, getChallengeAndSession, getExperiment, getExperiments, getListOfOpportunities, getListOfThumbnails, getPayload, getSatellite, getThumbnail, listFailure, listOpportunities, login, logout, newData, requestFailure, requestRejected, scheduleExperiment, unscheduleExperiment, and updateExperiment.

Inheritance diagram for xmlrpc_c::method:

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

- server/clientconnectivity.h
- server/mcsListener.cpp
- server/mccListener.cpp
3.2.92 XmlRpcClient Class Reference

#include <xmlRpcClient.h>

Inherited by MccClient, and McsClient.

Inheritance diagram for XmlRpcClient:

```
XmlRpcClient

# myCarriageParmPtr
# myClientPtr
# myUrl
+ XmlRpcClient()
+ XmlRpcClient()
+ ... 
+ erase()
+ exec()
+ fetch()
+ kill()
+ list()
+ McsClient
+ run()
+ schedule
+ store()
```

Public Member Functions

- **XmlRpcClient ()**
  Empty constructor of client object.

- **XmlRpcClient (const std::string &myUrl, const std::string &serverUrl)**
  Constructor of client object.

- **~XmlRpcClient ()**
  Destructor of client object.

Protected Attributes

- **xmlrpc_c::carriageParm_curl0 * myCarriageParmPtr**
  The parameters used for the connection.

- **xmlrpc_c::client_xml * myClientPtr**
  The actual XML-RPC client.

- **std::string myUrl**
  The URL that can be used to report.

Constructor & Destructor Documentation

XmlRpcClient (const std::string & myUrl, const std::string & serverUrl)
CHAPTER 3. COMPONENT DESCRIPTIONS

Parameters:

*myUrl*  The local URL the server can use for his response.

*serverUrl*  The URL for the server.

Precondition:

true

XmlRpcClient()

~XmlRpcClient()

Precondition:

true

Member Data Documentation

xmlrpc_c::carriageParm_curl0* myCarriageParmPtr [protected]

xmlrpc_c::client_xml* myClientPtr [protected]

std::string myUrl [protected]

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

- server/xmlRpcClient.h
- server/xmlRpcClient.cpp

3.2.93 XmlRpcController Class Reference

This class manages all connections with XML RPC servers.

#include <xmlRpcController.h>

Inherited by MccController, and McsController.
Inheritance diagram for XmlRpcController:

Public Member Functions

- std::string getHostname ()
  
  *Retrieve the local hostname from the config file 'server.icf'.*

- int removeBooking (Satellite *sat, int id)
  
  *Removes information of the request with identifier id.*

- ~XmlRpcController ()
  
  *Destructor of XML RPC controller object.*

Protected Member Functions

- SocketInfo getNewSocket ()
  
  *A function to get a socket bound to an unused port.*

Protected Attributes

- std::map< SatelliteID, std::map< int, int > > bookingIds
  
  *A map used for mapping server ids to bookingIds per satellite.*

- int lastUsed
  
  *The port that was last used for a XML RPC listener.*

- Scheduler * scheduler
Scheduler (p. 302) used for callbacks.

- std::map<int, int> timeouts
  A map used for mapping server bookingIds to timeouts per satellite.

Classes

- class XmlRpcControllerException
  Exception class thrown by the Controller (p. 170).

Constructor & Destructor Documentation

~XmlRpcController ()

Precondition:

true

Member Function Documentation

string getHostname ()

SocketInfo getNewSocket () [protected]

Precondition:

true

Returns:

A socket bound to a port and that port.

int removeBooking (Satellite * sat, int id)

Parameters:

sat Satellite (p. 293) to which the request belongs.

id The identifier of the request.

Member Data Documentation

std::map<SatelliteID, std::map<int, int>> bookingIds [protected]

int lastUsed [protected]

Scheduler* scheduler [protected]
std::map<int, int> timeouts [protected]
The documentation for this class was generated from the following files:

- server/xmlRpcController.h
- server/xmlRpcController.cpp

### 3.2.94 XmlRpcController::XmlRpcControllerException Class Reference

Exception class thrown by the Controller (p. 170).

```
#include <xmlRpcController.h>
```

Inherits `runtime_error`.

Inheritance diagram for XmlRpcController::XmlRpcControllerException:

```
+ XmlRpcControllerException
    + XmlRpcControllerException()
    + runtime_error
```

**Public Member Functions**

- `XmlRpcControllerException` (const std::string &arg)

**Constructor & Destructor Documentation**

```
XmlRpcControllerException (const std::string & arg) [inline]
```

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

- server/xmlRpcController.h

### 3.2.95 XmlRpcListener Class Reference

This class can be used to receive messages from a XML RPC server.

```
#include <xmlRpcListener.h>
```

Inherited by ClientConnectivity, MccListener, and McsListener[private].
Inheritance diagram for XmlRpcListener:

Public Member Functions

- void run ()
  
  Start listening for incoming messages.

- XmlRpcListener ()
  
  Constructor for XmlRpcListener (p. 328) object.

Protected Attributes

- xmlrpc_c::serverAbyss * myAbyssServerPtr
  
  The XML RPC server.

Constructor & Destructor Documentation

XmlRpcListener () [inline]

Member Function Documentation

void run ()

Precondition:

true

Postcondition:

The listeners is listening for incoming messages.

Member Data Documentation

xmlrpc_c::serverAbyss * myAbyssServerPtr [protected]

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

- server/xmlRpcListener.h
- server/xmlRpcListener.cpp
CHAPTER 3. COMPONENT DESCRIPTIONS

3.2.96 common/account.cpp File Reference

```
#include "account.h"
#include "connectionmanager.h"
```

Include dependency graph for account.cpp:

![Dependency Graph for account.cpp]

3.2.97 common/account.h File Reference

```
#include <string>
#include <stdexcept>
#include "defs.h"
```

Include dependency graph for account.h:

![Dependency Graph for account.h]

This graph shows which files directly or indirectly include this file:

![Dependency Graph for Include]

3.2.98 Classes

- class Account
  
    `std::exception`

- class Account::AccountException
Detailed Description

Author:
Stijn Stiefelhagen, Horus

The account class is an administration frontend for managing account objects.

3.2.98 common/adminlowlevel.h File Reference

```cpp
#include <string>
#include <stdexcept>
#include <vector>
#include <list>
#include "defs.h"
```

Include dependency graph for adminlowlevel.h:

This graph shows which files directly or indirectly include this file:
Classes

- class AdminLowlevel
  
  Class that presents an administration interface to the lower layers of the system.

- class AdminLowlevel::AdminLowlevelException
  
  Exception class thrown by AdminLowlevel (p. 126).

Detailed Description

Author:

Horus

Administration interface to the lower layers of the system.

\[3.2.99\] common/command.cpp File Reference

#include "command.h"
#include "connectionmanager.h"
#include <vector>

Include dependency graph for command.cpp:

\[\text{functions}\]

- std::vector< CommandID > getAllCommandIDs () throw (Command::CommandException)

  Gets the IDs of all commands in the database.

Function Documentation

std::vector<CommandID> getAllCommandIDs () throw (Command::CommandException)

Exceptions:

CommandException
Precondition:
true

Postcondition:
\{ c.Id | c in connection.Command \}

3.2.100 common/command.h File Reference

```c++
#include <string>
#include <stdexcept>
#include "defs.h"
#include "adminlowlevel.h"
```

Include dependency graph for command.h:

This graph shows which files directly or indirectly include this file:

Classes

- class Command
  < std::exception

- class Command::CommandException
  Exception class thrown by Command (p. 162).

Functions

- std::vector< CommandID > getAllCommandIDs () throw (Command::CommandException)
Gets the IDs of all commands in the database.

Detailed Description

Author:
Frank Koenders, Horus

The Command (p. 162) class is an administration frontend for managing command entities.

Function Documentation

\[
\text{std::vector<CommandID> getAllCommandIDs () throw (Command::CommandException)}
\]

Exceptions:

CommandException

Precondition:
true

Postcondition:
\{ c.Id | c in connection.Command \}

3.2.101 common/connectionmanager.cpp File Reference

#include "connectionmanager.h"
#include "databaseabstraction.h"

Include dependency graph for connectionmanager.cpp:

3.2.102 common/connectionmanager.h File Reference

#include "adminlowlevel.h"
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for connectionmanager.h:

This graph shows which files directly or indirectly include this file:

Classes

- class ConnectionManager

  *Provide a connection to the lower layer in order to abstract from the actual connection.*

Detailed Description

Author:

Carst Tankink, Horus

Provides a connection to the lower layer. For a server, this is the database abstraction. For the client, this is the server connectivity. This is an implementation of the Singleton pattern.

3.2.103 common/data.cpp File Reference

#include "data.h"

DDD 0.1 335
#include "connectionmanager.h"

Include dependency graph for data.cpp:

```
#include <vector>
#include <string>
#include "adminlowlevel.h"
#include <stdexcept>
```

## Functions

- `std::vector<DataID> getAllDataIDs () throw (Data::DataException)`

  *Gets the IDs of all data items in the database.*

### Function Documentation

```cpp
std::vector<DataID> getAllDataIDs () throw (Data::DataException)
```

**Exceptions:**

- `DataException`

**Precondition:**

`true`

**Postcondition:**

```cpp
{ d.Id | p in connection.Data }
```

### 3.2.104 common/data.h File Reference

```cpp
#include <vector>
#include <string>
#include "adminlowlevel.h"
#include <stdexcept>
```
Include dependency graph for data.h:

Classes

- class Data
  
  \textless\ text{std::exception} \text{ Administration frontend for managing data entities.}

- class Data::DataException
  
  Exception class thrown by Data (p. 199).

Functions

- std::vector\textless DataID \textgreater\ getAllDataIDs () throw (Data::DataException)
  
  Gets the IDs of all data items in the database.

Detailed Description

Author:

Frank Koenders, Horus
The data class is an administration frontend for managing data entities.

### Function Documentation

```cpp
std::vector<DataID> getAllDataIDs () throw (Data::DataException)
```

**Exceptions:**

*DataException*

**Precondition:**

`true`

**Postcondition:**

`{ d.Id | p in connection.Data }`

### 3.2.105 common/defs.h File Reference

```cpp
#include <string>
#include <vector>
```

Include dependency graph for `defs.h`:
This graph shows which files directly or indirectly include this file:
Classes

- class BookingInfo
  Specifies a booking.

- class CommandParam
  Parameters of a command which can be used in an experiment.

- class ExperimentCommand
  Parameters of a command as used in an experiment.

- class ExperimentPart
  Part of an experiment, as used on MCC/MCS and scheduler.

- class Interval
  Specifies an interval in time, to be used in a booking.

- union ParamTypes
  List of possible types for the parameters of a command.

- class QueueItem
  Specifies a queue item, as used in queues.

- class SingleLog
  Specifies a log item.

Typedefs

- typedef std::string AccountID
  Account (p. 114) identifier type.

- typedef int BookingID
  Booking (p. 150) identifier type.

- typedef std::vector< BookingInfo > BookingList
  List of bookings.

- typedef int CommandID
  Command (p. 162) identifier type.

- typedef int DataID
  Data (p. 199) identifier type.

- typedef std::vector< DataID > DataList
  List of data ids.

- typedef int ExperimentID
  Experiment (p. 221) identifier type.
CHAPTER 3. COMPONENT DESCRIPTIONS

- typedef std::vector<ExperimentPart> ExperimentParts
  List of experiment parts, for use in scheduler.

- typedef std::string GSSID

- typedef int ItemID
  (Queue (p. 283)) item identifier type

- typedef std::vector<CommandID> ListOfCommands
  List of all possible commands in an experiment.

- typedef std::vector<ExperimentCommand> ListOfExperimentCommands
  List of commands (part of an experiment).

- typedef int LogID
  log identifier type

- typedef int PayloadID
  Payload (p. 276) identifier type.

- typedef int SatelliteID
  Satellite (p. 293) identifier type.

Enumerations

- enum AccountType { Administrator, Observer, Scientist }
  Specifies the type of an account.

- enum Action {
  Exec, Fetch, Store, Erase,
  List, Schedule, Run, Kill,
  Unschedule }
  Specifies an action to be sent to the satellite.

- enum BookingStatus {
  Created, Pending, Cancelled, Accepted,
  Rejected, Failure, Running, Finished,
  Defined }
  Specifies the status of a booking.

- enum BookingType { Internal, External, UserRequested }
  Specifies the type of a booking.

- enum Direction { OneWay, TwoWay }
  Specifies the direction of a communication with a satellite.
• enum ExperimentStatus {
    Local, Uploaded, Executing, Executed,
    Scheduled }

    Specifies the status of an experiment.

• enum ParamType { Boolean, Double, Integer }

    Enumeration of the possible types of the parameters of a command.

Detailed Description

Author:

Horus

This file contains global definitions for IMSETY

Typedef Documentation

typedef std::string AccountID

typedef int BookingID

typedef std::vector<BookingInfo> BookingList

typedef int CommandID

typedef int DataID

typedef std::vector<DataID> DataList

typedef int ExperimentID

typedef std::vector<ExperimentPart> ExperimentParts

typedef std::string GSSID

GSS identifier type

typedef int ItemID

typedef std::vector<CommandID> ListOfCommands

typedef std::vector<ExperimentCommand> ListOfExperimentCommands
typedef int LogID

typedef int PayloadID

typedef int SatelliteID

Enumeration Type Documentation

enum AccountType

  Enumerator:

    Administrator
    Observer
    Scientist

enum Action

  Enumerator:

    Exec
    Fetch
    Store
    Erase
    List
    Schedule
    Run
    Kill
    Unschedule

enum BookingStatus

  Enumerator:

    Created
    Pending
    Cancelled
    Accepted
    Rejected
    Failure
    Running
    Finished
    Defined
enum BookingType
Enumerator:
    Internal
    External
    UserRequested

enum Direction
Enumerator:
    OneWay
    TwoWay

enum ExperimentStatus
Enumerator:
    Local
    Uploaded
    Executing
    Executed
    Scheduled

enum ParamType
Enumerator:
    Boolean
    Double
    Integer

3.2.106 common/experiment.cpp File Reference

#include "experiment.h"
#include "connectionmanager.h"

Include dependency graph for experiment.cpp:
CHAPTER 3. COMPONENT DESCRIPTIONS

8375 Functions

- std::vector<ExperimentID> getAllExperimentIDs () throw (Experiment::ExperimentException)

    Gets the IDs of all experiments in the database.

8380 Function Documentation

std::vector<ExperimentID> getAllExperimentIDs () throw (Experiment::ExperimentException)

Exceptions:

    ExperimentException

Precondition:

    true

Postcondition:

    { e.Id | e in connection.Experiment }

3.2.107 common/experiment.h File Reference

#include <string>
#include <stdexcept>
#include <vector>
#include <utility>
#include "defs.h"
#include "adminlowlevel.h"

Include dependency graph for experiment.h:
This graph shows which files directly or indirectly include this file:

![Graph showing file inclusion]

### Classes

- **class Experiment**

  \( <\text{std::utility}\) Administration frontend for managing experiment entities.

- **class Experiment::ExperimentException**

### Functions

- **std::vector<ExperimentID> getAllExperimentIDs() throw (Experiment::ExperimentException)**

  *Gets the IDs of all experiments in the database.*

### Detailed Description

**Author:**

Frank Koenders, Horus

The **Experiment** (p. 221) class is an administration frontend for managing experiment entities.

**Function Documentation**

**std::vector<ExperimentID> getAllExperimentIDs() throw (Experiment::ExperimentException)**

**Exceptions:**

*ExperimentException*

**Precondition:**

true

**Postcondition:**

\( \{ \text{e.Id} \mid \text{e in connection.Experiment} \} \)
3.2.108 common/log.cpp File Reference

```cpp
#include <iostream>
#include "log.h"
#include "connectionmanager.h"

Include dependency graph for log.cpp:

![Dependency Graph]

Functions

- `void addLog (int aTime, std::string aMessage, int aPriority, const SingleLog::Component &aSource, const ExperimentID &anExperimentID, const PayloadID &aPayloadID, const SatelliteID &aSatelliteID) throw (LogException)`
  Add a new log to the database.

- `void addLog (const SingleLog &aLog) throw (LogException)`
  Add a new log to the database.

- `void log (int aTime, std::string aMessage, int aPriority, const SingleLog::Component &aSource, const ExperimentID &anExperimentID, const PayloadID &aPayloadID, const SatelliteID &aSatelliteID) throw (LogException)`
  Add a new log to the database, or if the database is not working print to standard error.

- `void log (const SingleLog &aLog)`
  Add a new log to the database, or if the database is not working print to standard error.

- `SingleLog retrieveLog (LogID id) throw (LogException)`
  Retrieve the log with a certain identifier.

- `std::vector<LogID> retrieveLogs (int begintime, int endtime, const SingleLog::Component &source, int priority, const ExperimentID &anExperimentID, const PayloadID &aPayloadID, const SatelliteID &aSatelliteID) throw (LogException)`
  Return a list of LogIDs that are between begintime and endtime, are from component source and have a priority priority.

Function Documentation

`void addLog (int aTime, std::string aMessage, int aPriority, const SingleLog::Component &aSource, const ExperimentID &anExperimentID, const PayloadID &aPayloadID, const SatelliteID &aSatelliteID) throw (LogException)`

DDD 0.1 347
anExperimentID, aPayloadID and aSatelliteID can be set to -1 when the log does not apply to one of these.

Exceptions:

*LogException* (p. 247)

Parameters:

- **aTime** The time of the log.
- **aMessage** The message of the log.
- **aPriority** An (integer) priority of the message.
- **aSource** A Singelog::Component representing the source of the log.
- **anExperimentID** An ExperimentID representing the experiment the log belongs to (can be -1).
- **aPayloadID** An PayloadID representing the payload the log belongs to (can be -1).
- **aSatelliteID** A SatelliteID representing the satellite the log belongs to (can be -1).

Precondition:

true

Postcondition:

Log added to logs in the database.

```cpp
void addLog (const SingleLog & aLog) throw (LogException)
```

Exceptions:

*LogException* (p. 247)

Parameters:

- **aLog** The log entry to be added.

Precondition:

true

Postcondition:

Log added to logs in the database.

```cpp
void log (int aTime, std::string aMessage, int aPriority, const SingleLog::Component & aSource, const ExperimentID & anExperimentID, const PayloadID & aPayloadID, const SatelliteID & aSatelliteID)
```

anExperimentID, aPayloadID and aSatelliteID can be set to -1 when the log does not apply to one of these.

Parameters:

- **aTime** The time of the log.
CHAPTER 3. COMPONENT DESCRIPTIONS

**aMessage** The message of the log.

**aPriority** An (integer) priority of the message.

**aSource** A Singelog::Component representing the source of the log.

**anExperimentID** An ExperimentID representing the experiment the log belongs to (can be -1).

**aPayloadID** An PayloadID representing the payload the log belongs to (can be -1).

**aSatelliteID** A SatelliteID representing the satellite the log belongs to (can be -1).

**Precondition:**
true

**Postcondition:**
Log added to logs in the database, or printed to standard error.

```cpp
void log (const SingleLog & aLog)

Parameters:
aLog The log entry to be added.

Precondition:
true

Postcondition:
Log added to logs in the database, or printed to standard error.
```

```cpp
SingleLog retrieveLog (LogID id) throw (LogException)

Exceptions:
LogException (p. 247)

Parameters:
id The identifier of a certain log to be retrieved.

Precondition:
LogID > -1

Postcondition:
SingleLog.ID = LogID
```
std::vector<LogID> retrieveLogs (int begintime, int endtime, const SingleLog::Component & source, int priority, const ExperimentID & anExperimentID, const PayloadID & aPayloadID, const SatelliteID & aSatelliteID) throw (LogException)

Usage: to filter on certain parameters fill these in. The parameters source, priority, anExperimentID, aPayloadID, aSatellite can set to -1 when no filtering is required on one or more of these values.

Exceptions:

LogException (p. 247)

Parameters:

begintime Get logs starting from time begintime.
endtime Get logs ending at time endtime.
source Get logs which were made by the the component source.
priority Get logs with priority priority.
anExperimentID An experiment to which the log belongs to.
aPayloadID A payload to which the log belongs to.
aSatelliteID A satellite to which the log belongs to.

Precondition:

true

Postcondition:

All logIDs passing the filter.

3.2.109 common/log.h File Reference

#include "defs.h"
#include <vector>
#include <stdexcept>

Include dependency graph for log.h:
This graph shows which files directly or indirectly include this file:

```
common/log.h
common/log.cpp
server/controller.h
server/clientconnectivity.cpp
server/scheduler.cpp
server/mccController.cpp
server/mccListener.cpp
server/mcsController.cpp
server/mcsListener.cpp
server/xmlRpcUtils.h
server/controller.cpp
server/main.cpp
server/xmlRpcUtils.cpp
```

Classes

- class LogException
  
  `< std::exception`

Functions

- void addLog (int aTime, std::string aMessage, int aPriority, const SingleLog::Component &aSource, const ExperimentID &anExperimentID, const PayloadID &aPayloadID, const SatelliteID &aSatelliteID) throw (LogException)

  Add a new log to the database.

- void addLog (const SingleLog &aLog) throw (LogException)

  Add a new log to the database.

- void log (int aTime, std::string aMessage, int aPriority, const SingleLog::Component &aSource, const ExperimentID &anExperimentID, const PayloadID &aPayloadID, const SatelliteID &aSatelliteID)

  Add a new log to the database, or if the database is not working print to standard error.

- void log (const SingleLog &aLog)

  Add a new log to the database, or if the database is not working print to standard error.

- SingleLog retrieveLog (LogID id) throw (LogException)

  Retrieve the log with a certain identifier.

- std::vector< LogID > retrieveLogs (int begintime, int endtime, const SingleLog::Component &source, int priority, const ExperimentID &anExperimentID, const PayloadID &aPayloadID, const SatelliteID &aSatelliteID) throw (LogException)
Return a list of LogIDs that are between begintime and endtime, are from component source and have a priority priority.

Detailed Description
Author:
Stijn Stiefelhagen, Horus The log is an administration frontend for managing log objects.

Function Documentation

void addLog (int aTime, std::string aMessage, int aPriority, const SingleLog::Component & aSource, const ExperimentID & anExperimentID, const PayloadID & aPayloadID, const SatelliteID & aSatelliteID) throw (LogException)

anExperimentID, aPayloadID and aSatelliteID can be set to -1 when the log does not apply to one of these.

Exceptions:

LogException (p. 247)

Parameters:

aTime The time of the log.
aMessage The message of the log.
aPriority An (integer) priority of the message.
aSource A Singelog::Component representing the source of the log.
anExperimentID An ExperimentID representing the experiment the log belongs to (can be -1).
aPayloadID An PayloadID representing the payload the log belongs to (can be -1).
aSatelliteID A SatelliteID representing the satellite the log belongs to (can be -1).

Precondition:
true

Postcondition:
Log added to logs in the database.

void addLog (const SingleLog & aLog) throw (LogException)

Exceptions:

LogException (p. 247)

Parameters:

aLog The log entry to be added.

Precondition:
true
CHAPTER 3. COMPONENT DESCRIPTIONS

Postcondition:

Log added to logs in the database.

void log (int aTime, std::string aMessage, int aPriority, const SingleLog::Component & aSource, const ExperimentID & anExperimentID, const PayloadID & aPayloadID, const SatelliteID & aSatelliteID)

anExperimentID, aPayloadID and aSatelliteID can be set to -1 when the log does not apply to one of these.

Parameters:

- **aTime** The time of the log.
- **aMessage** The message of the log.
- **aPriority** An (integer) priority of the message.
- **aSource** A Singelog::Component representing the source of the log.
- **anExperimentID** An ExperimentID representing the experiment the log belongs to (can be -1).
- **aPayloadID** An PayloadID representing the payload the log belongs to (can be -1).
- **aSatelliteID** A SatelliteID representing the satellite the log belongs to (can be -1).

Precondition:

true

Postcondition:

Log added to logs in the database, or printed to standard error.

void log (const SingleLog & aLog)

Parameters:

- **aLog** The log entry to be added.

Precondition:

true

Postcondition:

Log added to logs in the database, or printed to standard error.

SingleLog retrieveLog (LogID id) throw (LogException)

Exceptions:

- **LogException** (p. 247)

Parameters:

- **id** The identifier of a certain log to be retrieved.
Precondition:

LogID > -1

Postcondition:

SingleLog.ID = LogID

\[\text{std::vector<LogID> retrieveLogs (int } \text{begintime, int } \text{endtime, const SingleLog::Component} \& \text{source, int } \text{priority, const ExperimentID} \& \text{anExperimentID, const PayloadID} \& \text{aPayloadID, const SatelliteID} \& \text{aSatelliteID) throw (LogException)}\]

Usage: to filter on certain parameters fill these in. The parameters source, priority, anExperimentID, aPayloadID, aSatellite can set to -1 when no filtering is required on one or more of these values.

Exceptions:

LogException (p. 247)

Parameters:

\text{begintime} Get logs starting from time \text{begintime}.

\text{endtime} Get logs ending at time \text{endtime}.

\text{source} Get logs which were made by the \text{source} component.

\text{priority} Get logs with priority \text{priority}.

\text{anExperimentID} An experiment to which the log belongs to.

\text{aPayloadID} A payload to which the log belongs to.

\text{aSatelliteID} A satellite to which the log belongs to.

Precondition:

true

Postcondition:

All logIDs passing the filter.

3.2.110 common/payload.cpp File Reference

#include "payload.h"
#include "connectionmanager.h"
Namespaces

- namespace std

Functions

- std::vector<PayloadID> getAllPayloadIDs() throw (Payload::PayloadException)

  Gets the IDs of all payloads in the database.

Function Documentation

std::vector<PayloadID> getAllPayloadIDs() throw (Payload::PayloadException)

Exceptions:

PayloadException

Precondition:

true

Postcondition:

{ p.Id | p in connection.Payload }

3.2.111  common/payload.h File Reference

#include <vector>
#include <string>
#include "defs.h"
#include <stdexcept>
Include dependency graph for payload.h:

This graph shows which files directly or indirectly include this file:

Classes

- class Payload
  
  Administration frontend for managing payload entities.

- class Payload::PayloadException

  Exception class thrown by Payload (p. 276).

Functions

- std::vector<PayloadID> getAllPayloadIDs () throw (Payload::PayloadException)

  Gets the IDs of all payloads in the database.

Detailed Description

Author:

Thijs Nugteren, Horus

The Payload (p. 276) class is an administration frontend for managing payload entities.

Function Documentation

std::vector<PayloadID> getAllPayloadIDs () throw (Payload::PayloadException)
CHAPTER 3. COMPONENT DESCRIPTIONS

Exceptions:

\textit{PayloadException}

Precondition:

true

Postcondition:

\{ p.Id \mid p \text{ in } connection.Payload \}

3.2.112 common/satellite.cpp File Reference

#include "satellite.h"
#include "connectionmanager.h"

Function Documentation

\texttt{std::vector<SatelliteID> getAllSatelliteIDs () throw (Satellite::SatelliteException)}

\textit{Gets the IDs of all satellites in the database.}

Function Documentaion

\texttt{std::vector<SatelliteID> getAllSatelliteIDs () throw (Satellite::SatelliteException)}

Exceptions:

\textit{SatelliteException}

Precondition:

true

Postcondition:

\{ s.Id \mid s \text{ in } connection.Satellite \}
### 3.2.1 File Reference

```cpp
#include <string>
#include <stdexcept>
#include "defs.h"
```

Include dependency graph for satellite.h:

This graph shows which files directly or indirectly include this file:

#### Classes

- class **Satellite**
  
  `< std::exception Administration frontend for managing satellite entities.`

- class **Satellite::SatelliteException**
  
  Exception class thrown by **Satellite** (p. 293).

#### Functions

- std::vector< **SatelliteID** > **getAllSatelliteIDs** () throw (**Satellite::SatelliteException**)
  
  **Gets the IDs of all satellites in the database.**

#### Detailed Description

**Author:**

Carst Tankink, Horus
The Satellite (p. 293) class is an administration frontend for managing satellite entities.

Function Documentation

std::vector<SatelliteID> getAllSatelliteIDs () throw (Satellite::SatelliteException)

Exceptions:
SatelliteException

Precondition:
true

Postcondition:
{ s.Id | s in connection.Satellite }

3.2.114 common/sha2.cpp File Reference

#include "sha2.h"
#include <iostream>

Include dependency graph for sha2.cpp:

Variables

- const sha_word32 K1_0_TO_19 = 0x5a827999UL
- const sha_word32 K1_20_TO_39 = 0x6ed9eba1UL
- const sha_word32 K1_40_TO_59 = 0x8f1bbec6UL
- const sha_word32 K1_60_TO_79 = 0xc9a62c1d6UL
- static const sha_word32 K256 [64]
- static const sha_word64 K512 [80]
- static const sha_word32 sha1_initial_hash_value [5]
- static const sha_word32 sha224_initial_hash_value [8]
- static const sha_word32 sha256_initial_hash_value [8]
- static const sha_word64 sha384_initial_hash_value [8]
- static const sha_word64 sha512_initial_hash_value [8]
- static const char * sha_hex_digits = "0123456789abcdef"

Variable Documentation

const sha_word32 K1_0_TO_19 = 0x5a827999UL
const sha_word32 K1_20_TO_39 = 0x6ed9eba1UL
const sha_word32 K1_40_TO_59 = 0x8f1bbdcdUL

const sha_word32 K1_60_TO_79 = 0xca62c1d6UL

const sha_word32 K256[64] [static]
Initial value:

{  0x428a2f98UL, 0x71374491UL, 0xb5c0fbcfUL, 0xe9b5dba5UL,  
  0x3956c25bUL, 0x59f111f1UL, 0xaabbce45UL, 0x4bb16af2UL,  
  0x8070d83bUL, 0x1f25dc87UL, 0x976f8180UL, 0x8206f298UL,  
  0x806bd9b8UL, 0x243185beUL, 0xm6040f94UL, 0x2253f113UL,  
  0x90befffaUL, 0xa4506cebUL, 0x5be0cd19UL, 0x360DD10.1  
}

const sha_word64 K512[80] [static]

const sha_word32 sha1_initial_hash_value[5] [static]
Initial value:

{  0x67452301UL, 0xefcdab89UL, 0x98badcfeUL, 0x10325476UL,  
   0xc3d2e1f0UL  
}

const sha_word32 sha224_initial_hash_value[8] [static]
Initial value:

{  0xc1059ed8UL, 0x367cd507UL, 0x3070dd17UL, 0xf70e5939UL,  
   0xffc00b31UL, 0x68581511UL, 0x64f98fa7UL, 0x5be0cd19UL  
}

const sha_word32 sha256_initial_hash_value[8] [static]
Initial value:

{  0x6a09e667UL, 0xbb67ae85UL, 0x3c6ef372UL, 0xa54ff53aUL,  
   0x510e527fUL, 0x9b05688cUL, 0x1f83d9abUL, 0x5be0cd19UL  
}
const sha_word64 sha384_initial_hash_value[8] [static]
Initial value:

```
{ 0xcbbb9d5dc1059ed8ULL, 0x629a292a367cd507ULL,
  0x9159015a3070dd17ULL, 0x152fecd8f70e5939ULL,
  0x67332667ffc00b31ULL, 0x8eb44a8768581511ULL,
  0xdb0c2e0d64f98fa7ULL, 0x47b5481dbefa4fa4ULL }
```

const sha_word64 sha512_initial_hash_value[8] [static]
Initial value:

```
{ 0x6a09e667f3bcc908ULL, 0xbb67ae8584caa73bULL,
  0x3c6ef372fe94f82bULL, 0xa54ff53a5f1d36f1ULL,
  0x510e527fade682d1ULL, 0x9b05688c2b3e6c1fULL,
  0x1f83d9abfb41bd6bULL, 0x5be0cd19137e2179ULL }
```

const char* sha_hex_digits = "0123456789abcdef" [static]

3.2.115 common/sha2.h File Reference

```
#include <string>
#include <stdexcept>
```

Include dependency graph for sha2.h:

This graph shows which files directly or indirectly include this file:

Classes

- class sha2
- struct sha2::SHA_CTX
CHAPTER 3. COMPONENT DESCRIPTIONS

Typedefs

- typedef unsigned char sha_byte
- typedef unsigned int sha_word32
- typedef unsigned long long sha_word64

Variables

- const sha_word32 SHA1_DIGESTC_LENGTH = 20
- const sha_word32 SHA1_DIGESTC_STRING_LENGTH = (SHA1_DIGESTC_LENGTH ∗ 2 + 1)
- const sha_word32 SHA224_DIGESTC_LENGTH = 28
- const sha_word32 SHA224_DIGESTC_STRING_LENGTH = (SHA224_DIGESTC_LENGTH ∗ 2 + 1)
- const sha_word32 SHA256_DIGESTC_LENGTH = 32
- const sha_word32 SHA256_DIGESTC_STRING_LENGTH = (SHA256_DIGESTC_LENGTH ∗ 2 + 1)
- const sha_word32 SHA384_DIGESTC_LENGTH = 48
- const sha_word32 SHA384_DIGESTC_STRING_LENGTH = (SHA384_DIGESTC_LENGTH ∗ 2 + 1)
- const sha_word32 SHA512_DIGESTC_LENGTH = 64
- const sha_word32 SHA512_DIGESTC_STRING_LENGTH = (SHA512_DIGESTC_LENGTH ∗ 2 + 1)

Typedef Documentation

typedef unsigned char sha_byte

typedef unsigned int sha_word32

typedef unsigned long long sha_word64

Variable Documentation

const sha_word32 SHA1_DIGESTC_LENGTH = 20

const sha_word32 SHA1_DIGESTC_STRING_LENGTH = (SHA1_DIGESTC_LENGTH ∗ 2 + 1)

const sha_word32 SHA224_DIGESTC_LENGTH = 28

const sha_word32 SHA224_DIGESTC_STRING_LENGTH = (SHA224_DIGESTC_LENGTH ∗ 2 + 1)

const sha_word32 SHA256_DIGESTC_LENGTH = 32
const sha_word32 SHA256_DIGESTC_STRING_LENGTH = (SHA256_DIGESTC_LENGTH * 2 + 1)

const sha_word32 SHA384_DIGESTC_LENGTH = 48

const sha_word32 SHA384_DIGESTC_STRING_LENGTH = (SHA384_DIGESTC_LENGTH * 2 + 1)

const sha_word32 SHA512_DIGESTC_LENGTH = 64

const sha_word32 SHA512_DIGESTC_STRING_LENGTH = (SHA512_DIGESTC_LENGTH * 2 + 1)

### 3.2.116 common/utils.h File Reference

```cpp
#include <sstream>
#include <string>
#include "sha2.h"
```

Include dependency graph for utils.h:

This graph shows which files directly or indirectly include this file:

#### Functions

- **std::string hash (const std::string data)**
  
  *Hash a string.*

- **std::string hash (const unsigned char *data, const int length)**
  
  *Hash a character stream taking the first length characters into account.*

- **std::string toString (const int i)**
  
  *Convert integer to string.*
Detailed Description

Author:
Jeroen Keiren, Horus.

This file provides some convenience functions that can be used in the rest of the code.

Function Documentation

```
std::string hash (const std::string data) [inline]
```

Parameters:
- **data** String that needs to be hashed.

Returns:
SHA256 hash of data.

```
std::string hash (const unsigned char * data, const int length) [inline]
```

Parameters:
- **data** Data (p. 199) that needs to be hashed.
- **length** The amount of data being used to produce the hash.

Returns:
SHA256 hash of the first length characters of data.

```
std::string toString (const int i) [inline]
```

Parameters:
- **i** The integer to be converted.

Returns:
The string representation of i.

3.2.117 server/booking.cpp File Reference

```cpp
#include "booking.h"
#include "connectionmanager.h"
#include "experiment.h"
#include "payload.h"
```
Include dependency graph for booking.cpp:

```
server/booking.cpp
booking.h
connectionmanager.h
experiment.h
payload.h
string
stdexcept
defs.h
vector
adminlowlevel.h
list
utility
```

### Functions

- `std::vector<BookingID> getAllBookingIDs (BookingType type)`
  
  *get the associated external bookings of an experiment*

- `std::vector<BookingID> getExternalBookingsOfExperiment (ExperimentID id)`
  
  *get the first booking from now.*

- `BookingID getInternalBooking (SatelliteID satelliteID)`
  
  *Get the bookingId for the satellite, return -1 if not exists.*

- `std::vector<BookingID> getTimeBookingIDs (int time)`
  
  *Retrieve all bookings for a given time.*

### Function Documentation

#### `std::vector<BookingID> getAllBookingIDs (BookingType type)`

- **Parameters:**
  
  - `id` The experimentid for which the bookings are to be fetched

- **Precondition:**
  
  - true

- **Returns:**
  
  - `{b.id | b in connection.Booking /\ b.getExperimentID = id /\ b.type = External}`
CHAPTER 3. COMPONENT DESCRIPTIONS

**BookingID getFirstBooking ()**

**Precondition:**

true

**Returns:**

b.identifier with b in connection.Booking and Forall (b’ in connection.Booking :: b’.starttime >= b.starttime)

**BookingID getInternalBooking (SatelliteID satelliteID)**

**Parameters:**

* satelliteID The satelliteID for which we want the internal booking.

**Precondition:**

true

**Returns:**

BookingID of internal booking with status either ”created” or ”pending” or ”accepted” if the internal booking for satellite exists else -1.

**std::vector<BookingID> getTimeBookingIDs (int time)**

**Parameters:**

* time Time for which the bookings have to be retrieved

**Returns:**

All bookings for during a given time

### 3.2.118 server/booking.h File Reference

```
#include <string>
#include <stdexcept>
#include "defs.h"
```

Include dependency graph for booking.h:

This graph shows which files directly or indirectly include this file:
CHAPTER 3. COMPONENT DESCRIPTIONS

Classes

- class Booking
- class Booking::BookingException

Functions

- std::vector< BookingID > getAllBookingIDs (BookingType &type)
  
  Gets the IDs of all bookings in the database.

- std::vector< BookingID > getExternalBookingsOfExperiment (ExperimentID id)
  
  get the associated external bookings of an experiment

- BookingID getFirstBooking ()
  
  < Typedef for SatelliteID. Gets the first booking from now.

- BookingID getInternalBooking (SatelliteID satelliteID)
  
  Get the bookingId for the satellite, return -1 if not exists.

- std::vector< BookingID > getTimeBookingIDs (int time)
  
  Retreive all bookings for a given time.

Detailed Description

Author:

Thijs Nugteren, Horus

The booking class is a frontend for managing booking entities in the server

Function Documentation

std::vector<BookingID> getAllBookingIDs (BookingType & type)

Parameters:

  type a booking type

Precondition:

  true

Postcondition:

  \{ b.Id | b in connection.Booking /\ b.type = type \}
std::vector<BookingID> getExternalBookingsOfExperiment (ExperimentID id)

Parameters:

id The experimentid for which the bookings are to be fetched

Precondition:

true

Returns:

{b.id | b in connection.Booking \ b.getExperimentID = id \ b.type = External}

BookingID getFirstBooking ()

Precondition:

true

Returns:

b.identifier with b in connection.Booking and Forall (b' in connection.Booking :: b'.starttime \== b.starttime)

BookingID getInternalBooking (SatelliteID satelliteID)

Parameters:

satelliteID The satelliteID for which we want the internal booking.

Precondition:

true

Returns:

BookingID of internal booking with status either "created" or "pending" or "accepted" if the internal booking for satellite exists else -1.

std::vector<BookingID> getTimeBookingIDs (int time)

Parameters:

time Time for which the bookings have to be retrieved

Returns:

All bookings for during a given time
3.2.119  server/clientconnectivity.cpp File Reference

```c++
#include "clientconnectivity.h"
#include "controller.h"
#include "log.h"
#include "xmlRpcUtils.h"
#include <string>
#include <vector>
#include <utility>
```
Include dependency graph for clientconnectivity.cpp:
3.2.120 server/clientconnectivity.h File Reference

#include <xmlrpc-c/registry.hpp>
#include <xmlrpc-c/server_abyss.hpp>
#include "xmlRpcListener.h"

Include dependency graph for clientconnectivity.h:

This graph shows which files directly or indirectly include this file:

Classes:

- class changePassword
  Changes the password of an account.

- class ClientConnectivity
- class createExperiment
  Creates a new experiment for a payload with payloadID and with a name.

- class getChallengeAndSession
  Returns a challenge and a session, to be used by the user to identify himself to the server.

- class getExperiment
  Get experiment by identifier.

- class getExperiments
  Gets all satellites, payloads and experiments associated with an account.

- class getListOfOpportunities
  Returns a list of opportunities for an experiment with experimentID.

- class getListOfThumbnails
  Requests a list of thumbnails.

- class getPayload
  Get payload by identifier.

- class getSatellite
CHAPTER 3. COMPONENT DESCRIPTIONS

Get satellite by identifier.

- class getThumbnail
  Get the next observation thumbnail for an experiment.

- class login
  Login remote procedure call.

- class logout
  Logout remote procedure call.

- class scheduleExperiment
  Requests to schedule an experiment.

- class unscheduleExperiment
  Requests to unschedule an experiment.

- class updateExperiment
  Updates the experiment on the server.

Detailed Description
Author:
Carst Tankink, Horus

Contains a class used for communicating with one client.

3.2.121 server/controller.cpp File Reference

#include <pthread.h>
#include "controller.h"
#include "rnd.h"
#include "utils.h"
#include <unistd.h>
#include <unistd.h>
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for controller.cpp:
CHAPTER 3. COMPONENT DESCRIPTIONS

Defines

- #define KEEPALIVE_TIME 3600

Functions

- void * awaitListOfOpportunities (void *args)
- void * awaitNewData (void *args)
- void * safetyThread (void *args)

Define Documentation

#define KEEPALIVE_TIME 3600

Function Documentation

void* awaitListOfOpportunities (void * args)

void* awaitNewData (void * args)

void* safetyThread (void * args)

Todo

Currently nothing prevents two clients from editing the same payload, which could cause inconsistency problems, for example because of one scientist overriding another’s work. This can be overcome by implementing a locking mechanism on payloads.

3.2.122 server/controller.h File Reference

#include <map>

#include <utility>

#include <vector>

#include <stdexcept>

#include <string>

#include <time.h>

#include "account.h"

#include "command.h"

#include "defs.h"

#include "experiment.h"

#include "log.h"

#include "payload.h"

#include "satellite.h"

#include "scheduler.h"
Include dependency graph for controller.h:
This graph shows which files directly or indirectly include this file:

Classes

- class Controller
- class Controller::CallbackHandleOpportunities
  
  Callback handle for requesting list of opportunities.

- class Controller::ControllerException
  
  Exception thrown by the Controller (p. 170) class.

- class Controller::ExperimentInfo
  
  Class used in the return value of getExperimentIDs.

- class Controller::PayloadInfo
  
  Class used in the return value of getPayloadIDs.

- class Controller::SessionInfo
  
  Simple structure that is used to keep the information belonging to a session.

- class Controller::ThreadArgumentInt
  
  Structure used to pass controller and id into a thread.

- class Controller::ThreadArgumentStringInt
  
  Structure used to pass controller, id and session into a thread.

- class Controller::ThreadingInformation
  
  Administration for threads.

Detailed Description

Author:

Jeroen Keiren, Horus

The Controller (p. 170) is the main logic provider of the IMSETY server.
3.2.123 server/databaseabstraction.cpp File Reference

```cpp
#include "databaseabstraction.h"
#include <iostream>
#include <exception>
#include <unistd.h>
#include <fstream>
```

Include dependency graph for databaseabstraction.cpp:

```
Defines

- `#define INITIAL_SLEEP_TIME 100000`

Define Documentation

```
#define INITIAL_SLEEP_TIME 100000
```

3.2.124 server/databaseabstraction.h File Reference

```cpp
#include "adminlowlevel.h"
#include "defs.h"
#include <mysql++/mysql++.h>
#include <string>
#include <vector>
```

```cpp
#include <iostream>
#include <exception>
#include <unistd.h>
#include <fstream>
```
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for databaseabstraction.h:

This graph shows which files directly or indirectly include this file:

Classes

- **class** DatabaseAbstraction

  < Type definition for SatelliteID.>

- **class** DatabaseAbstraction::DatabaseException

3.2.125 server/main.cpp File Reference

```cpp
#include "clientconnectivity.h"
#include "controller.h"
#include "version.h"
#include <iostream>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/errno.h>
```
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for main.cpp:
CHAPTER 3. COMPONENT DESCRIPTIONS

Functions

- int main (int argc, char *argv[])
Classes

- class MccClient
  
  This class can be used to send messages to a MCC.

Detailed Description

Author:

Joeri de Ruiter, Horus

This file contains the class used for connection to a MCC.

3.2.128 server/mccController.cpp File Reference

#include "mccController.h"
#include "mccListener.h"
#include "xmlRpcListener.h"
#include "log.h"
#include <time.h>
#include <iostream>

Include dependency graph for mccController.cpp:
CHAPTER 3. COMPONENT DESCRIPTIONS

Defines

- #define SLEEP 10
- #define TIMEOUT_RESPONSE 60

Functions

- void * runMccTimeouts (void *ptr)

Define Documentation

# define SLEEP 10

# define TIMEOUT_RESPONSE 60

Function Documentation

void* runMccTimeouts (void *ptr)

3.2.129 server/mccController.h File Reference

#include <map>
#include "satellite.h"
#include "xmlRpcController.h"
#include "mccClient.h"
#include "scheduler.h"
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for mccController.h:

This graph shows which files directly or indirectly include this file:

Classes

- **class MccController**
  
  *This class manages all connections with the MCCs.*

- **class MccInfo**
  
  *This class contains information for communication with a MCC.*
Detailed Description

Author:
Joeri de Ruiter, Horus

This file contains the class used for managing connections with the MCCs.

3.2.130 server/mccGeneral.cpp File Reference

#include <map>
#include "mccGeneral.h"

Include dependency graph for mccGeneral.cpp:

Functions

- xmlrpc_c::value_struct bookingInfoFromClass (const BookingInfo & booking)
  
  Convert a BookingInfo (p. 157) to a XML RPC struct.

- BookingInfo bookingInfoToClass (const xmlrpc_c::value_struct & booking)
  
  Convert a XML RPC struct to a BookingInfo (p. 157).

- BookingList bookingListFromXML (const xmlrpc_c::value_array & bookings)
  
  Convert a XML RPC struct to a BookingList.

- xmlrpc_c::value_struct intervalFromClass (const Interval & period)
  
  Convert a Period to a XML RPC struct.

Function Documentation

xmlrpc_c::value_struct bookingInfoFromClass (const BookingInfo & booking)

Parameters:

booking The booking to be converted.

Precondition:

true

Returns:

The XML RPC struct that represents booking.
CHAPTER 3. COMPONENT DESCRIPTIONS

BookingInfo bookingInfoToClass (const xmlrpc_c::value_struct & booking)

Parameters:
 booking The booking to be converted.

Precondition:
 true

Returns:
 The BookingInfo (p. 157) that represents booking.

BookingList bookingListFromXML (const xmlrpc_c::value_array & bookings)

Parameters:
 bookings The bookinglist to be converted.

Precondition:
 true

Returns:
 The BookingList that represents bookings.

xmlrpc_c::value_struct intervalFromClass (const Interval & period)

Parameters:
 period The period to be converted.

Precondition:
 true

Returns:
 The XML RPC struct that represents period.

3.2.131 server/mccGeneral.h File Reference

#include <xmlrpc-c/base.hpp>
#include "defs.h"

Include dependency graph for mccGeneral.h:
Functions

- **xmlrpc_c::value_struct bookingInfoFromClass (const BookingInfo &booking)**  
  Convert a *BookingInfo* (p. 157) to a XML RPC struct.

- **BookingInfo bookingInfoToClass (const xmlrpc_c::value_struct &booking)**  
  Convert a XML RPC struct to a *BookingInfo* (p. 157).

- **BookingList bookingListFromXML (const xmlrpc_c::value_array &bookings)**  
  Convert a XML RPC struct to a BookingList.

- **xmlrpc_c::value_struct intervalFromClass (const Interval &period)**  
  Convert a *Period to a XML RPC struct.*

Detailed Description

Author:

Joeri de Ruiter, Horus

This file contains functions that are used by different MCC components.

Function Documentation

**xmlrpc_c::value_struct bookingInfoFromClass (const BookingInfo & booking)**

Parameters:

- **booking** The booking to be converted.

Precondition:

true

Returns:

The XML RPC struct that represents booking.
CHAPTER 3. COMPONENT DESCRIPTIONS

BookingInfo bookingInfoToClass (const xmlrpc_c::value_struct & booking)

Parameters:

booking The booking to be converted.

Precondition:
true

Returns:
The BookingInfo (p. 157) that represents booking.

BookingList bookingListFromXML (const xmlrpc_c::value_array & bookings)

Parameters:

bookings The bookinglist to be converted.

Precondition:
true

Returns:
The BookingList that represents bookings.

xmlrpc_c::value_struct intervalFromClass (const Interval & period)

Parameters:

period The period to be converted.

Precondition:
true

Returns:
The XML RPC struct that represents period.

3.2.132 server/mccListener.cpp File Reference

#include "mccListener.h"
#include "log.h"
#include <iostream>

DDD 0.1 387
Include dependency graph for mccListener.cpp:

```
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for mccListener.cpp:

```

Classes

- class booked
- class cancelled
- class listFailure
- class listOfOpportunities
- class requestFailure
- class requestRejected

3.2.133 server/mccListener.h File Reference

```cpp
#include <xmlrpc-c/registry.hpp>

#include <xmlrpc-c/server_abyss.hpp>
#include "mccGeneral.h"
#include "mccController.h"
#include "xmlRpcListener.h"
```
This graph shows which files directly or indirectly include this file:

Classes

• class MccListener

  This class can be used to receive messages from a MCC.

Detailed Description

Author:

Joeri de Ruiter, Horus

This file contains the class used for connection from a MCC.
CHAPTER 3. COMPONENT DESCRIPTIONS

3.2.134 server/mcsClient.cpp File Reference

```cpp
#include <xmlrpc-c/girerr.hpp>
#include <xmlrpc-c/base.hpp>
#include <xmlrpc-c/client_simple.hpp>
#include "mcsClient.h"
```

Include dependency graph for mcsClient.cpp:

3.2.135 server/mcsClient.h File Reference

```cpp
#include <xmlrpc-c/client.hpp>
#include "xmlRpcClient.h"
#include "mcsGeneral.h"
```

Include dependency graph for mcsClient.h:

This graph shows which files directly or indirectly include this file:

Classes

- class **McsClient**

  This class can be used to send messages to a MCS.
CHAPTER 3. COMPONENT DESCRIPTIONS

Detailed Description
Author: Joeri de Ruiter, Horus

This file contains the class used for connection to a MCS.

3.2.136 server/mcsController.cpp File Reference

#include "mcsController.h"
#include "mcsListener.h"
#include "xmlRpcListener.h"
#include "log.h"
#include <time.h>
#include <iostream>

Include dependency graph for mcsController.cpp:

Defines

- #define SLEEP 10
- #define TIMEOUT_RESPONSE 60
CHAPTER 3. COMPONENT DESCRIPTIONS

Functions

- void * runMcsTimeouts (void *ptr)

Define Documentation

#define SLEEP 10

#define TIMEOUT_RESPONSE 60

Function Documentation

void* runMcsTimeouts (void *ptr)

3.2.137 server/mcsController.h File Reference

#include <map>
#include "satellite.h"
#include "xmlRpcController.h"
#include "mcsClient.h"
#include "mcsGeneral.h"
#include "scheduler.h"
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for mcsController.h:

This graph shows which files directly or indirectly include this file:

Classes

- class **McsController**
  
  *This class manages all connections with the MCSs.*

- class **McsInfo**
  
  *This class contains information for communication with an MCS.*

Detailed Description

Author:

    Joeri de Ruiter, Horus
This file contains the class used for managing connections with the MCSs.

### 3.2.138 server/mcsGeneral.cpp File Reference

```cpp
#include <map>
#include "mcsGeneral.h"
```

Include dependency graph for mcsGeneral.cpp:

#### Functions

- `xmlrpc_c::value_array experimentInfoToXML (ExperimentParts experiment)`
  
  Convert an `ExperimentInfo` to an XMLRPC array.

- `xmlrpc_c::value_struct experimentPartToXML (const ExperimentPart &part)`
  
  Convert an `ExperimentPart` (p. 233) to an XMLRPC struct.

#### Function Documentation

**xmlrpc_c::value_array experimentInfoToXML (ExperimentParts experiment)**

**Parameters:**

- `experiment` The `ExperimentInfo` to be converted.

**Returns:**

The converted XMLRPC array.

**xmlrpc_c::value_struct experimentPartToXML (const ExperimentPart & part)**

**Parameters:**

- `part` The `ExperimentPart` (p. 233) to be converted.

**Returns:**

The converted XMLRPC struct.

### 3.2.139 server/mcsGeneral.h File Reference

```cpp
#include <xmlrpc-c/base.hpp>
#include "defs.h"
```
CHAPTER 3. COMPONENT DESCRIPTIONS

Include dependency graph for mcsGeneral.h:

```
This graph shows which files directly or indirectly include this file:
```

Functions

- `xmlrpc_c::value_array experimentInfoToXML (ExperimentParts experiment)`
  
  Convert an `ExperimentInfo` to an XMLRPC array.

- `xmlrpc_c::value_struct experimentPartToXML (const ExperimentPart &part)`
  
  Convert an `ExperimentPart` (p. 233) to an XMLRPC struct.

Detailed Description

Author:

Joeri de Ruiter, Horus

This file contains general functions for MCS communication.

Function Documentation

`xmlrpc_c::value_array experimentInfoToXML (ExperimentParts experiment)`

Parameters:

- `experiment` The `ExperimentInfo` to be converted.

Returns:

The converted XMLRPC array.

`xmlrpc_c::value_struct experimentPartToXML (const ExperimentPart & part)`

Parameters:

- `part` The `ExperimentPart` (p. 233) to be converted.

Returns:

The converted XMLRPC struct.
3.2.140 server/mcsListener.cpp File Reference

```
#include <iostream>
#include "mcsListener.h"
#include "log.h"
```

Include dependency graph for mcsListener.cpp:

Classes

- class acknowledged
- class failed
- class newData

3.2.141 server/mcsListener.h File Reference

```
#include "mcsController.h"
#include "xmlRpcListener.h"
```
Include dependency graph for mcsListener.h:

This graph shows which files directly or indirectly include this file:

Classes

- class McsListener

This class can be used to receive messages from a MCS.

Detailed Description

Author:

Joeri de Ruiter, Horus
CHAPTER 3. COMPONENT DESCRIPTIONS

This file contains the class used for connection from a MCS.

3.2.142 server/queue.cpp File Reference

#include "queue.h"
#include "connectionmanager.h"

Include dependency graph for queue.cpp:

3.2.143 server/queue.h File Reference

#include <string>
#include <list>
#include <stdexcept>
#include "defs.h"

Include dependency graph for queue.h:
CHAPTER 3. COMPONENT DESCRIPTIONS

This graph shows which files directly or indirectly include this file:

Classes

- class Queue
  
  < Typedef for SatelliteID.

- class Queue::QueueException

Detailed Description

Author:

Frank Koenders, Horus

The queue class is a queue used by the scheduler to manage the uploading to satellites.

3.2.144 server/rnd.h File Reference

```plaintext
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>
```

Include dependency graph for rnd.h:
This graph shows which files directly or indirectly include this file:

![Graph showing dependencies between files](image)

### Functions

- `unsigned char * rnd (int n)`
  
  *Generate n bytes of random data.*

- `void seed_rand (void)`
  
  *Seed the random number generator.*

### Function Documentation

`unsigned char* rnd (int n)`

`void seed_rand (void)`

### 3.2.145 server/scheduler.cpp File Reference

```c
#include "scheduler.h"
#include "booking.h"
#include "experiment.h"
#include "payload.h"
#include "log.h"
#include "command.h"
#include "controller.h"
#include "utils.h"
#include <time.h>
#include <list>
#include <sstream>
#include <string>
#include <iostream>
#include "mccController.h"
#include "mcsController.h"
#include <cassert.h>
```
Include dependency graph for scheduler.cpp:
CHAPTER 3. COMPONENT DESCRIPTIONS

Defines

- \#define SLEEP_TIME 30

Functions

- void * runSatellite (void *arg)
  
  The thread that checks a satellite.

- void * runScheduler (void *arg)
  
  The thread that checks the scheduler periodically.

- void * runUpdateThread (void *arg)

Define Documentation

\#define SLEEP_TIME 30

Function Documentation

void* runSatellite (void *arg)

Parameters:

arg Thread argument.

void* runScheduler (void *arg)

Parameters:

arg Thread argument.

void* runUpdateThread (void *arg)

3.2.146 server/scheduler.h File Reference

#include <string>

#include <map>

#include <pthread.h>

#include "defs.h"

#include "satellite.h"

#include "data.h"

#include "queue.h"
#include "connectionmanager.h"
#include "xmlRpcController.h"
#include <stdexcept>

Include dependency graph for scheduler.h:

This graph shows which files directly or indirectly include this file:

Classes

• class Queues
Class that contains an internal and an external queue.

- **class SchedInfo**
  
  A class that contains information needed by the thread that checks a satellite.

- **class Scheduler**
- **class Scheduler::SchedulerException**
  
  Exception thrown by the scheduler.

### Functions

- **void * runSatellite (void *arg)**
  
  The thread that checks a satellite.

- **void * runScheduler (void *arg)**
  
  The thread that checks the scheduler periodically.

### Detailed Description

**Author:**

Stijn Stiefelhagen, Horus The scheduler class interacts with the MCC and MCS.

**Function Documentation**

**void* runSatellite (void * arg)**

**Parameters:**

- **arg** Thread argument.

**void* runScheduler (void * arg)**

**Parameters:**

- **arg** Thread argument.

### 3.2.147 server/version.h File Reference

This graph shows which files directly or indirectly include this file:

```
server/version.h  server/main.cpp
```

**Defines**

- `#define IMSETY_VERSION "0.2.1"`
CHAPTER 3. COMPONENT DESCRIPTIONS

Define Documentation

#define IMSETY_VERSION "0.2.1"

3.2.148 server/xmlRpcClient.cpp File Reference

#include <xmlrpc-c/base.hpp>
#include "xmlRpcClient.h"

Include dependency graph for xmlRpcClient.cpp:

This graph shows which files directly or indirectly include this file:

Classes

• class XmlRpcClient

3.2.150 server/xmlRpcController.cpp File Reference

#include <sys/socket.h>
#include <unistd.h>
#include <netinet/in.h>
#include <sys/errno.h>
#include <fstream>

DD 0.1 405
#include <iostream>
#include "scheduler.h"
#include "satellite.h"
#include "xmlRpcController.h"
#include "xmlRpcListener.h"

Include dependency graph for xmlRpcController.cpp:

Functions

- void * runListener (void *arg)

  Function to be used in a thread to run a XmlRpcListener (p. 328).

Function Documentation

void* runListener (void * arg)

3.2.151 server/xmlRpcController.h File Reference

#include <map>
#include <pthread.h>
#include <stdexcept>
#include "defs.h"

Include dependency graph for xmlRpcController.h:

This graph shows which files directly or indirectly include this file:

Classes

- **struct SocketInfo**
  
  Convenience struct.

- **class XmlRpcController**
  
  This class manages all connections with XML RPC servers.

- **class XmlRpcController::XmlRpcControllerException**
  
  Exception class thrown by the Controller (p. 170).

Functions

- **void *runListener** (void *arg)
  
  Function to be used in a thread to run a XmlRpcListener (p. 328).
Detailed Description

Author:
Joeri de Ruiter, Horus

This file contains the class used for managing connections with XML RPC servers.

Function Documentation

```c
void* runListener (void * arg)
```

3.2.152 server/xmlRpcListener.cpp File Reference

```c
#include <iostream>
#include "xmlRpcListener.h"
```

Include dependency graph for xmlRpcListener.cpp:

3.2.153 server/xmlRpcListener.h File Reference

```c
#include <xmlrpc-c/base.hpp>
#include <xmlrpc-c/registry.hpp>
#include <xmlrpc-c/server_abyss.hpp>
```

Include dependency graph for xmlRpcListener.h:
Classes

- class XmlRpcListener

   This class can be used to receive messages from a XML RPC server.

Detailed Description

Author:

Joeri de Ruiter, Horus

This file contains the class used for connection from a XML RPC server.

3.2.154 server/xmlRpcUtils.cpp File Reference

#include <map>
#include "xmlRpcUtils.h"
Include dependency graph for xmlRpcUtils.cpp:
FUNCTIONS

- \texttt{xmlrpc\_c::value\_struct commandToXMLStruct (Command c)}
  
  Translates \texttt{Command} (p. 162) object \texttt{c} into an XML-RPC struct containing the information of that object.

- \texttt{xmlrpc\_c::value\_struct expCommandToXMLStruct (ExperimentCommand c)}
  
  Translates \texttt{ExperimentCommand} (p. 232) object \texttt{c} into an XML-RPC struct containing the information of that object.

- \texttt{std::string experimentStatusString (ExperimentStatus es)}
  
  Converts an \texttt{ExperimentStatus} to a STL string.

- \texttt{xmlrpc\_c::value\_struct experimentToXMLStruct (Experiment experiment, Controller *controller, std::string session)}
  
  Translates \texttt{Experiment} (p. 221) object \texttt{experiment} into an XML-RPC struct containing the information of that object.

- \texttt{std::string paramTypeToString (ParamType t)}
  
  Converts a parameter type to a STL string.

- \texttt{xmlrpc\_c::value\_struct payloadToXMLStruct (Payload payload)}
  
  Translates \texttt{Payload} (p. 276) object \texttt{payload} into an XML-RPC struct containing the information of that object.

- \texttt{xmlrpc\_c::value\_struct satelliteToXMLStruct (Satellite satellite)}
  
  Translates \texttt{Satellite} (p. 293) object \texttt{satellite} into an XML-RPC struct containing the information of that object.

FUNCTION DOCUMENTATION

\texttt{xmlrpc\_c::value\_struct commandToXMLStruct (Command c)}

Parameters:

- \texttt{c} The command to be converted.

Precondition:

- true

Postcondition:

- An XML-RPC struct containing the information contained in \texttt{c}.

\texttt{xmlrpc\_c::value\_struct expCommandToXMLStruct (ExperimentCommand c)}

Parameters:

- \texttt{c} The command to be converted.
Precondition:
true

Postcondition:
An XML-RPC struct containing the information contained in \( c \).

```cpp
std::string experimentStatusString (ExperimentStatus es)
```

**Parameters:**

\( es \) The ExperimentStatus to be converted to a string.

**Precondition:**
true

**Returns:**
The string corresponding to the enum denominator of \( es \).

```cpp
xmlrpc_c::value_struct experimentToXMLStruct (Experiment experiment, Controller * controller, std::string session)
```

**Parameters:**

\( experiment \) The experiment to be converted.
\( controller \) The controller from which possible commands are requested.
\( session \) The session identifier used to identify the requesting user with the controller.

**Precondition:**
true

**Postcondition:**
An XML-RPC struct containing the information contained in \( payload \).

```cpp
std::string paramTypeToString (ParamType t)
```

**Parameters:**

\( t \) The ParamType to be converted to a string.

**Precondition:**
true

**Returns:**
The string corresponding to the enum denominator of \( t \).
xmlrpc_c::value_struct payloadToXMLStruct (Payload *payload)

Parameters:

payload The payload to be converted.

Precondition:

true

Postcondition:

An XML-RPC struct containing the information contained in payload.

xmlrpc_c::value_struct satelliteToXMLStruct (Satellite *satellite)

Parameters:

satellite The satellite to be converted.

Precondition:

true

Postcondition:

An XML-RPC struct containing the information contained in satellite.

3.2.155 server/xmlRpcUtils.h File Reference

#include <xmlrpc-c/base.hpp>
#include <string>
#include "command.h"
#include "controller.h"
#include "defs.h"
#include "experiment.h"
#include "payload.h"
#include "satellite.h"
Include dependency graph for xmlRpcUtils.h:
CHAPTER 3. COMPONENT DESCRIPTIONS

This graph shows which files directly or indirectly include this file:

```
server/xmlRpcUtils.h
server/clientconnectivity.cpp
server/xmlRpcUtils.cpp
```

Functions

- `xmlrpc_c::value_struct commandToXMLStruct (Command c)`
  
  Translates `Command` (p. 162) object `c` into an XML-RPC struct containing the information of that object.

- `xmlrpc_c::value_struct expCommandToXMLStruct (ExperimentCommand c)`
  
  Translates `ExperimentCommand` (p. 232) object `c` into an XML-RPC struct containing the information of that object.

- `std::string experimentStatusString (ExperimentStatus es)`
  
  Converts an `ExperimentStatus` to a STL string.

- `xmlrpc_c::value_struct experimentToXMLStruct (Experiment experiment, Controller ∗controller, std::string session)`
  
  Translates `Experiment` (p. 221) object `experiment` into an XML-RPC struct containing the information of that object.

- `std::string paramTypeToString (ParamType t)`
  
  Converts a parameter type to a STL string.

- `xmlrpc_c::value_struct payloadToXMLStruct (Payload payload)`
  
  Translates `Payload` (p. 276) object `payload` into an XML-RPC struct containing the information of that object.

- `xmlrpc_c::value_struct satelliteToXMLStruct (Satellite satellite)`
  
  Translates `Satellite` (p. 293) object `satellite` into an XML-RPC struct containing the information of that object.

Function Documentation

`xmlrpc_c::value_struct commandToXMLStruct (Command c)`

Parameters:

- `c` The command to be converted.

Precondition:

- true

Postcondition:

- An XML-RPC struct containing the information contained in `c`. 
xmlrpc_c::value_struct expCommandToXMLStruct (ExperimentCommand c)

Parameters:
  c  The command to be converted.

Precondition:
  true

Postcondition:
  An XML-RPC struct containing the information contained in c.

std::string experimentStatusString (ExperimentStatus es)

Parameters:
  es  The ExperimentStatus to be converted to a string.

Precondition:
  true

Returns:
  The string corresponding to the enum denominator of es.

xmlrpc_c::value_struct experimentToXMLStruct (Experiment experiment, Controller *controller, std::string session)

Parameters:
  experiment  The experiment to be converted.
  controller  The controller from which possible commands are requested.
  session     The session identifier used to identify the requesting user with the controller.

Precondition:
  true

Postcondition:
  An XML-RPC struct containing the information contained in payload.

std::string paramTypeToString (ParamType t)

Parameters:
  t  The ParamType to be converted to a string.

Precondition:
  true

Returns:
  The string corresponding to the enum denominator of t.
xmlrpc_c::value_struct payloadToXMLStruct (Payload payload)

Parameters:

payload The payload to be converted.

Precondition:

true

Postcondition:

An XML-RPC struct containing the information contained in payload.

xmlrpc_c::value_struct satelliteToXMLStruct (Satellite satellite)

Parameters:

satellite The satellite to be converted.

Precondition:

true

Postcondition:

An XML-RPC struct containing the information contained in satellite.

3.2.156 Todo List

Member Controller::addAccountToPayload (p. 176) Implement using Payload (p. 276) class.

Member Controller::addCommand (p. 177) Implement using Command (p. 162) class.

Member Controller::callbackExperimentUploadFailure (p. 177) This is currently unused, but could facilitate a negative answer to the client.

Member Controller::createAccount (p. 178) Implement using Account (p. 114) class.

Member Controller::createPayload (p. 179) Implement using Payload (p. 276) class.

Member Controller::createSatellite (p. 179) Implement using Satellite (p. 293) class.

Member Controller::deleteExperiment (p. 180) Implement deletion of experiment using the Experiment (p. 221) class.

Member Controller::getChallenge (p. 180) we need real random data.
CHAPTER 3. COMPONENT DESCRIPTIONS

Member Controller::getDatum (p. 181) Implement by checking the user has rights to the experiment the data belongs to. Then get the data and instantiate it. Throw an exception if no such data exists.

Member Controller::getLog (p. 183) Implement retrieval of logs using the Log class.

Member Controller::hasHighResolution (p. 184) Implement using Data (p. 199) class.

Member Controller::markForDeletion (p. 186) Implement marking for deletion of data using the Data (p. 199) class.

Member Controller::markForHighResolutionDownload (p. 186) Implement using Data (p. 199) class.

Member Controller::registerAccount (p. 187) Implement using Account (p. 114) class.

Member Controller::registerPayload (p. 187) Implement using Payload (p. 276) class.

Member Controller::registerSatellite (p. 187) Implement using Satellite (p. 293) class.

Member Controller::removeAccount (p. 187) Implement using Account (p. 114) class.

Member Controller::removeAccountFromPayload (p. 188) Implement using Payload (p. 276) class.

Member Controller::removeCommand (p. 188) Implement using Command (p. 162) class.

Member Controller::removePayload (p. 188) Implement using Payload (p. 276) class.

Member Controller::removeSatellite (p. 189) Implement using Satellite (p. 293) class.

Member Controller::sendCommand (p. 189) Implement by checking for the appropriate rights, and using the executeCommand function from the scheduler.

Member Controller::unregisterAccount (p. 190) Implement using Account (p. 114) class.

Member Controller::unregisterPayload (p. 190) Implement using Payload (p. 276) class.
CHAPTER 3. COMPONENT DESCRIPTIONS

Member Controller::unregisterSatellite (p. 190) Implement using Satellite (p. 293) class.

Member Controller::unscheduleExperiment (p. 191) Check when unscheduling may happen!

Member Controller::updateAccount (p. 191) Implement using Account (p. 114) class.

Member Controller::updateCommand (p. 191) Implement using Command (p. 162) class.

Member Controller::updatePayload (p. 192) Implement using Payload (p. 276) class.

Member Controller::updateSatellite (p. 193) Implement using Satellite (p. 293) class.

Member Data::~Data (p. 203) Clean up dataItem and thumbnail? What if this is copy-constructed?

Member DatabaseAbstraction::newQuery (p. 217) const Try to clean up this code as it should not be needed for the above logic it merely fixes a warning

Member DatabaseAbstraction::removeExperiment (p. 217) const Implement removal of experiment by simply executing an appropriate query.

Member DatabaseAbstraction::retrieveData (p. 218) const Implement retrieval of high resolution data

Member DatabaseAbstraction::updateData (p. 219) const Implement saving dataItem

Member DatabaseAbstraction::updateData (p. 219) const Implement update query

Member Scheduler::callbackBooking (p. 305) Here the controller could be informed that these experiments could not be uploaded in time by means of a callback. The controller will then be able to perform some actions such as informing the client.

Member Scheduler::callbackListOfOpportunities (p. 306) Here the controller could be informed that these experiments could not be uploaded in time by means of a callback. The controller will then be able to perform some actions such as informing the client.
**Member Scheduler::callbackTimeoutMCC** (p. 307) Implement: MCC informs scheduler of a Time out on a certain booking after which the controller can try to request to booking again.

**Member Scheduler::callbackTimeoutMCS** (p. 307) Implement: MCS informs scheduler of a Time out on a certain booking after which the controller can try to request to booking again.

**Member Scheduler::downloadHighResolutionData** (p. 309) Implementation of downloading high resolution data is not done.

**Member Scheduler::isWindowOfOpportunity** (p. 311) Implementation of this function has not been done. It could be used by the scheduler to find out whether there is a window of opportunity.

**Member Scheduler::processQueue** (p. 311) Add time constraint, so if something goes wrong the thread will stop eventually.

**Member Scheduler::processSatellite** (p. 312) Here the controller could be informed that these items have not been processed by means of a callback. The controller will then be able to perform some actions such as informing the client.

**Member safetyThread** (p. 374) Currently observing two experiments from the same client in parallel poses problems because the requestThumbnail call is blocking. This way, if no data is received from one experiment, the controller will block on this, and therefore the data of the other experiment will not be forwarded to the client either. This problem can be overcome by making each requestThumbnail call a separate thread.

**Member safetyThread** (p. 374) Currently nothing prevents two clients from editing the same payload, which could cause inconsistency problems, for example because of one scientist overriding another’s work. This can be overcome by implementing a locking mechanism on payloads.
Chapter 4

Build procedure

This chapter contains all information needed to build the IMSETY server and client, and their build time dependencies. Information concerning the run time dependencies can be found in section 4.1.

First the dependencies will be described in section 4.1. These will be described shortly giving the minimal information needed to get IMSETY running. More information about the dependencies and their possibilities can be found in their documentation, which can be found within the packages or on the websites.

Section 4.2 explains the procedure for building the IMSETY server. After which the build procedure for the IMSETY client is described in section 4.3.

4.1 Dependencies

The server has the following build time dependencies:

- MySQL++ (library), at least version 2.0.
- Xmlrpc-c (library), at least version 1.06.03.
- SCons (build environment), at least version 0.96.
- CppUnit (test framework), at least version 1.12.
- MySQL, at least version 5.

The stubs have the following extra dependencies:

- Apache, at least version 2.
- PHP, at least version 5.1.

These package are known by most of the operating system’s package managers. This provides the easiest way to install. Pay attention that the development editions of the libraries are needed, normally distinguishable by a ’-dev’ or ’-devel’ suffix in the package name. If the default installations do not work out, or if they are not provided by the package manager, install these packages according to the descriptions of the following sections.

Build environment, test framework and libraries for the client are all provided by the Qt framework. This can also be found in many package managers but manual installation instructions have
been included for completeness. For the administration front end the client has the following dependencies:

- Apache, at least version 2.
- PHP, at least version 5.1.
- MySQL, at least version 5.

4.1.1 MySQL++

MySQL++ is a dependency for the server. It can be found at http://www.tangentsoft.net/mysql++/ and http://www.mysql.org/downloads/api-mysql++.html. We require a minimum version of 2.0. To compile it execute the following commands:

```bash
# wget http://www.tangentsoft.net/mysql++/releases/mysql++-VERSION.tar.gz
# tar -zxvf mysql++-VERSION.tar.gz
# cd mysql++-VERSION
# ./configure --enable-exceptions --includedir=/usr/include
# make
```

Now it can be installed by executing the following command as root:

```bash
# make install
```

4.1.2 Xmlrpc-c

Xmlrpc-c is a dependency for the server. It can be found at http://xmlrpc-c.sourceforge.net/. We require at least version 1.06.03. Xmlrpc-c has the following dependencies:

- libxml2-dev
- libcurl3-dev

To compile Xmlrpc-c execute the following commands:

```bash
# wget http://surfnet.dl.sourceforge.net/sourceforge/xmlrpc-c/xmlrpc-c-VERSION.tgz
# tar -zxvf xmlrpc-c-VERSION.tgz
# cd xmlrpc-c-VERSION
# ./configure --disable-wininet-client --enable-libxml2-backend --enable-abyss-threads --enable-curl-client --prefix=/usr
# make
```

Now it can be installed by executing the following command as root:

```bash
# make install
```

4.1.3 Qt

Qt is a dependency for the client. It can be found at http://www.trolltech.com/. We require at least version 3.2.
CHAPTER 4. BUILD PROCEDURE

4.1.4 SCons

SCons is a build time dependency for the server. It provides the build system to compile the sources. It can be found at http://www.scons.org. We require at least version 0.96.

4.1.5 CppUnit

CppUnit is the framework that we use for testing our software. It can be found at http://cppunit.sourceforge.net. We require at least version 1.12.

4.1.6 MySQL

MySQL is the database server that is used by the server, the administration and the stubs. It can be found at http://www.mysql.org. We require at least version 5.

4.1.7 Apache

Apache is the web server on which the stubs and administration run. It can be found at http://www.apache.org. We require at least version 2.

4.1.8 PHP

PHP is the programming language that is used for both the administration and the stubs. It can be found at http://www.php.net. We require at least version 5.1.

4.2 Server

Go to the folder where you have placed the source code for IMSETY. Then execute the following commands:

# cd src/server
# scons

To test the code you can execute the following command:

# scons test

Before you can run the server you will have to update the server.icf configuration file to reflect the environment. Furthermore the MySQL database has to be initialized with the imsety.sql, which contains the database structure.

You can now run the server by executing the command:

# ./server
CHAPTER 4. BUILD PROCEDURE

4.3 Client

4.3.1 Linux, Mac OS X and Solaris

Go to the folder where you have placed the source code for IMSETY. Then execute the following commands:

```
# cd src/client
# qmake
# make
```

You can now run the client by executing:

```
# ./client
```

4.3.2 Windows

First you have to download Qt from the website http://www.trolltech.com/. When you have installed Qt you have to start the Qt command Prompt. Then go to the directory where the source code for the client is. Then compile the client using the following commands:

```
qmake
make
```

The executable is now placed in the subdirectory release. When opening it from the Qt Command Prompt you do not need to have dlls in the folder. When you want to be able to execute it outside the Qt command prompt you have to copy the following dlls to the directory:

- mingwm10.dll (from the MinGW bin directory)
- QtCore4.dll (from the Qt bin directory)
- QtGui4.dll (from the Qt bin directory)
- QtNetwork4.dll (from the Qt bin directory)
- QtXml4.dll (from the Qt bin directory)

You now do not need to have Qt installed anymore. You can copy the executable and the dlls to for example another computer without Qt and you can run it there.
Appendix A

Source code listings

The source code is distributed over three directories, client for the client code, and common and server for the server code. The configuration item lists can be found in:

- **client** section 3.1.7
- **common** section 3.2.8
- **server** section 3.2.9
Appendix B

Requirements traceability matrix

Here functional Software Requirements from the SRD [5] are traced to components from the Detailed Design.

<table>
<thead>
<tr>
<th>SRFUR</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRFUR1</td>
<td>Account</td>
</tr>
<tr>
<td>SRFUR2</td>
<td>Account</td>
</tr>
<tr>
<td>SRFUR3</td>
<td>Account, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR4</td>
<td>Account, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR5</td>
<td>Account, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR6</td>
<td>Account, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR7</td>
<td>Account, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR8</td>
<td>Account, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR9</td>
<td>Account, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR10</td>
<td>Account, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR11</td>
<td>Account, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR12</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR13</td>
<td>Satellite, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR14</td>
<td>Satellite, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR15</td>
<td>Satellite, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR16</td>
<td>Satellite, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR17</td>
<td>Payload, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR18</td>
<td>Payload, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR19</td>
<td>Payload, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR20</td>
<td>Payload, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR21</td>
<td>Payload, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR22</td>
<td>Payload, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR23</td>
<td>Command, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR24</td>
<td>Command, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR25</td>
<td>Command, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR26</td>
<td>Experiment, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR27</td>
<td>Experiment, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR28</td>
<td>Experiment, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR29</td>
<td>Experiment, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR30</td>
<td>Experiment, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR31</td>
<td>Experiment, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR32</td>
<td>Experiment, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR33</td>
<td>Experiment, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR34</td>
<td>Experiment, DatabaseAbstraction</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>SRFUR35</td>
<td>Experiment, Controller, Scheduler</td>
</tr>
<tr>
<td>SRFUR36</td>
<td>Queue</td>
</tr>
<tr>
<td>SRFUR37</td>
<td>Queue, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR38</td>
<td>Queue, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR39</td>
<td>Booking</td>
</tr>
<tr>
<td>SRFUR40</td>
<td>Booking</td>
</tr>
<tr>
<td>SRFUR41</td>
<td>Booking, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR42</td>
<td>Data</td>
</tr>
<tr>
<td>SRFUR43</td>
<td>Data</td>
</tr>
<tr>
<td>SRFUR44</td>
<td>Data, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR45</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR46</td>
<td>Data, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR47</td>
<td>Thumbnail</td>
</tr>
<tr>
<td>SRFUR48</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR49</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR50</td>
<td>Thumbnail</td>
</tr>
<tr>
<td>SRFUR51</td>
<td>Data</td>
</tr>
<tr>
<td>SRFUR52</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR53</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR54</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR55</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR56</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR57</td>
<td>ObservationView</td>
</tr>
<tr>
<td>SRFUR58</td>
<td>ObservationView</td>
</tr>
<tr>
<td>SRFUR59</td>
<td>ObservationData</td>
</tr>
<tr>
<td>SRFUR60</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR61</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR62</td>
<td>ObservationView</td>
</tr>
<tr>
<td>SRFUR63</td>
<td>ObservationView</td>
</tr>
<tr>
<td>SRFUR64</td>
<td>ObservationView</td>
</tr>
<tr>
<td>SRFUR65</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR66</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR67</td>
<td>Controller</td>
</tr>
<tr>
<td>SRFUR68</td>
<td>Controller</td>
</tr>
<tr>
<td>SRFUR69</td>
<td>Log</td>
</tr>
<tr>
<td>SRFUR70</td>
<td>Log, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR71</td>
<td>Log, DatabaseAbstraction</td>
</tr>
<tr>
<td>SRFUR72</td>
<td>MCS</td>
</tr>
<tr>
<td>SRFUR73</td>
<td>MCS</td>
</tr>
<tr>
<td>SRFUR74</td>
<td>MCS</td>
</tr>
<tr>
<td>SRFUR75</td>
<td>MCS</td>
</tr>
<tr>
<td>SRFUR76</td>
<td>MCS</td>
</tr>
<tr>
<td>SRFUR77</td>
<td>MCS</td>
</tr>
<tr>
<td>SRFUR78</td>
<td>MCS</td>
</tr>
<tr>
<td>SRFUR79</td>
<td>MCS</td>
</tr>
<tr>
<td>SRFUR80</td>
<td>MCS</td>
</tr>
<tr>
<td>SRFUR81</td>
<td>MCC</td>
</tr>
<tr>
<td>SRFUR82</td>
<td>MCC</td>
</tr>
<tr>
<td>SRFUR83</td>
<td>MCC</td>
</tr>
<tr>
<td>SRFUR84</td>
<td>MCC</td>
</tr>
</tbody>
</table>
## APPENDIX B. REQUIREMENTS TRACEABILITY MATRIX

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRFUR85</td>
<td>Controller</td>
</tr>
<tr>
<td>SRFUR86</td>
<td>Controller</td>
</tr>
<tr>
<td>SRFUR87</td>
<td>Thumbnail, ObservationView</td>
</tr>
<tr>
<td>SRFUR88</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR89</td>
<td>ObservationView</td>
</tr>
<tr>
<td>SRFUR90</td>
<td>Controller</td>
</tr>
<tr>
<td>SRFUR91</td>
<td>Controller</td>
</tr>
<tr>
<td>SRFUR92</td>
<td>Controller</td>
</tr>
<tr>
<td>SRFUR93</td>
<td>Not implemented</td>
</tr>
<tr>
<td>SRFUR94</td>
<td>Administration</td>
</tr>
<tr>
<td>SRFUR95</td>
<td>ExperimentDialog</td>
</tr>
<tr>
<td>SRFUR96</td>
<td>ExperimentDialog</td>
</tr>
<tr>
<td>SRFUR97</td>
<td>Mainview, ObservationView</td>
</tr>
<tr>
<td>SRFUR98</td>
<td>Payload</td>
</tr>
<tr>
<td>SRFUR99</td>
<td>MCS, Scheduler</td>
</tr>
<tr>
<td>SRFUR100</td>
<td>Controller, Scheduler, MCS</td>
</tr>
<tr>
<td>SRFUR101</td>
<td>ObservationView</td>
</tr>
<tr>
<td>SRFUR102</td>
<td>Log</td>
</tr>
<tr>
<td>SRFUR103</td>
<td>Log</td>
</tr>
</tbody>
</table>
Index

~Account
Account, 117
~AdminLowlevel
AdminLowlevel, 131
~Booking
Booking, 153
~ClientConnectivity
ClientConnectivity, 161
~Command
Command, 164
~Commands
Commands, 12
~CommandsEditDialog
CommandsEditDialog, 16
~CommunicationLogsDialog
CommunicationLogsDialog, 17
~Controller
Controller, 176
~CreateExperimentDialog
CreateExperimentDialog, 19
~Data
Data, 203
~DatabaseAbstraction
DatabaseAbstraction, 215
~Experiment
Experiment, 224
~ExperimentDialog
ExperimentDialog, 25
~LoginDialog
LoginDialog, 31
~MainWindow
MainWindow, 36
~ManageObservers
ManageObservers, 39
~ManagePayloadsDialog
ManagePayloadsDialog, 40
~ManageSatellitesDialog
ManageSatellitesDialog, 42
~ManageUsersDialog
ManageUsersDialog, 43
~MccListener
MccListener, 259
~MesListener
MesListener, 274
~MultiHttp
MultiHttp, 44
~ObservationDataDialog
ObservationDataDialog, 47
~ObservationView
ObservationView, 49
~Observer
Observer, 52
~Payload
Payload, 278
~Preferences
Preferences, 55
~Queue
Queue, 285
~Satellite
Satellite, 295
~Scheduler
Scheduler, 305
~SelectObserveDialog
SelectObserveDialog, 61
~ServerConnectivity
ServerConnectivity, 67
~UserModel
UserModel, 77
~UserSelectionWidget
UserSelectionWidget, 78
~XmlRpcClient
XmlRpcClient, 325
~XmlRpcController
XmlRpcController, 327

abort

MultiHttp, 44
Accepted
defs.h, 343
Account, 114
~Account, 117
Account, 116, 117
accountExists, 118
addExperiment, 118
addPayload, 118
eMail, 124
fullname, 124
getMail, 118
getFullName, 119
getID, 119
getPassword, 119
getType, 119
hasRightsToExperiment, 120
hasRightsToPayload, 120
hasRightsToSatellite, 120
identifier, 124
instantiate, 121
isRegistered, 121
needs_storage, 124
needs_update, 124
password, 124
registerAccount, 121
registered, 124
removeExperiment, 121
removePayload, 122
setEmail, 122
setFullName, 123
setPassword, 123
setType, 123
setUpdate, 123
store, 123
type, 124
_unregisterAccount, 124
Account::AccountException, 125
AccountException, 125
accountAddExperiment
AdminLowlevel, 132
DatabaseAbstraction, 215
accountAddPayload
AdminLowlevel, 132
DatabaseAbstraction, 215
Account::AccountException, 125
accountExists
Account, 118
AdminLowlevel, 132
DatabaseAbstraction, 215
accountHasRightsToExperiment
AdminLowlevel, 132
DatabaseAbstraction, 215
accountHasRightsToPayload
AdminLowlevel, 133
DatabaseAbstraction, 215
accountHasRightsToSatellite
AdminLowlevel, 133
DatabaseAbstraction, 215
AccountID
defs.h, 342
accountID
Controller::SessionInfo, 196
accountRemoveExperiment
AdminLowlevel, 133
DatabaseAbstraction, 215
accountRemovePayload
AdminLowlevel, 134
DatabaseAbstraction, 215
AccountType
defs.h, 343
acknowledged, 125
acknowledged, 126
ccontroller, 126
eexecute, 126
sat, 126
Action
defs.h, 343
action
QueueItem, 289
addAccountToPayload
Controller, 176
addCommand
Commands, 12
Controller, 176
Experiment, 225
addExperiment
Account, 118
ADDINCI28
sha2, 315
addLog
AdminLowlevel, 134
DatabaseAbstraction, 215
log.cpp, 347, 348
log.h, 352
addParameter
Commands, 12
addPayload
Account, 118
addTimedCommand
ExperimentDialog, 25
addUser
ManageObservers, 39
Administrator
defs.h, 343
AdminLowlevel, 126
~AdminLowlevel, 131
accountAddExperiment, 132
accountAddPayload, 132
accountExists, 132
accountHasRightsToExperiment, 132
accountHasRightsToPayload, 133
accountHasRightsToSatellite, 133
accountRemoveExperiment, 133
accountRemovePayload, 134
addLog, 134
getAllBookingIDs, 134
getAllCommandIDs, 135
getAllDataIDs, 135
getAllExperimentIDs, 135
getAllPayloadIDs, 135
<table>
<thead>
<tr>
<th>Function/Class/Definition</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>getAllSatelliteIDs, getAvailableSatellites, getDataList</td>
<td>135</td>
</tr>
<tr>
<td>getExperimentAtTime, getExternalBookingsOfExperiment, getFirstBooking, getInternalBooking, getPayloadExperiments</td>
<td>136</td>
</tr>
<tr>
<td>getPayloadListOfCommands, getQueue</td>
<td>138</td>
</tr>
<tr>
<td>getthumbnailList, removeExperiment, removeQueue, removeQueueItem</td>
<td>139</td>
</tr>
<tr>
<td>retrieveAccount, retrieveBooking, retrieveCommand, retrieveData</td>
<td>140</td>
</tr>
<tr>
<td>retrieveExperiment, retrieveLog, retrieveLogs</td>
<td>142</td>
</tr>
<tr>
<td>retrievePayload, retrieveQueueItem, retrieveSatellite, updateAccount,</td>
<td>144</td>
</tr>
<tr>
<td>updateBooking, updateCommand, updateData, updateCommand</td>
<td>145</td>
</tr>
<tr>
<td>updateexperiment, updatepayload, updatequeueitem, updateSatellite, updateData</td>
<td>145</td>
</tr>
<tr>
<td>AdminLowlevel::AdminLowlevelException</td>
<td>148</td>
</tr>
<tr>
<td>AdminLowlevelException</td>
<td>148</td>
</tr>
<tr>
<td>AdminLowlevelException</td>
<td>148</td>
</tr>
<tr>
<td>booking.cpp</td>
<td>148</td>
</tr>
<tr>
<td>ServerConnectivity, 67</td>
<td>67</td>
</tr>
<tr>
<td>ServerConnectivity, 74</td>
<td>74</td>
</tr>
<tr>
<td>AllThumbnailsRequest</td>
<td>67</td>
</tr>
<tr>
<td>Scheduler, 305</td>
<td>305</td>
</tr>
<tr>
<td>addObserver, ObservationView, 49</td>
<td>49</td>
</tr>
<tr>
<td>Preferences, 55</td>
<td>55</td>
</tr>
<tr>
<td>awaitListOpportunities, controller.cpp, 374</td>
<td>374</td>
</tr>
<tr>
<td>awaitNewData, controller.cpp, 374</td>
<td>374</td>
</tr>
</tbody>
</table>

**Bookmarks:**

- AdminLowlevel::AdminLowlevelException
- AdminLowlevel::AdminLowlevelException
- Booking
- booking.cpp
- ServerConnectivity
- Scheduler
- Preferences
- awaitListOpportunities
- controller.cpp
- awaitNewData
- controller.cpp

**Bracketed Numbers:**

- 10100
- 10105
- 10110
- 10115
- 10120
- 10125
- 10130
- 10135
- 10140
- 10145
- 10150
- 10155
- 10160
- 10165
- 10170
- 10175
- 10180
- 10185
- 10190
- 10195
- 10200

**Referenced Classes:**

- ParamTypes
- bitcount
- sha2::SHA_CTX
- booked
- controller
- execute
- sat
- Booking
- Booking::BookingException
- booking.cpp
- booking.h

**Referenced Functions:**

- getAllSatelliteIDs
- getAvailableSatellites
- getDataList
- getExperimentAtTime
- getExternalBookingsOfExperiment
- getFirstBooking
- getInternalBooking
- getPayloadExperiments
- getPayloadListOfCommands
- getQueue
- getThumbnailList
- getTimeBookingIDs
- removeExperiment
- removeQueue
- removeQueueItem
- retrieveAccount
- retrieveBooking
- retrieveCommand
- retrieveData
- retrieveExperiment
- retrieveLog
- retrieveLogs
- retrievePayload
- retrieveQueueItem
- retrieveSatellite
- updateAccount
- updateBooking
- updateCommand
- updateData
- updateExperiment
- updatePayload
- updateQueueItem
- updateSatellite

**Referenced Classes:**

- Booking
- Booking::BookingException
- AdminLowlevel::AdminLowlevelException

**Referenced Functions:**

- ParamTypes
- bitcount
- sha2::SHA_CTX
- booked
- controller
- execute
- sat
- Booking
- Booking::BookingException
- booking.cpp
- booking.h

**Referenced Classes:**

- ParamTypes
- bitcount
- sha2::SHA_CTX
- booked
- controller
- execute
- sat
- Booking
- Booking::BookingException
- booking.cpp
- booking.h
**defs.h**, 342
bookingIds
MccController, 257
McsController, 271
XmlRpcController, 327
BookingInfo, 157
gssid, 158
operator==, 158
period, 158
remarks, 158
twoway, 158
bookingInfoFromClass
mccGeneral.cpp, 384
mccGeneral.h, 386
bookingInfoToClass
mccGeneral.cpp, 384
mccGeneral.h, 386
BookingList
defs.h, 342
bookingListFromXML
mccGeneral.cpp, 385
mccGeneral.h, 387
bookings
Controller::CallbackHandleOpportunities
BookingStatus
defs.h, 343
BookingType
defs.h, 343
Boolean
defs.h, 344
buffer
sha2::SHA_CTX, 318
callbackAcknowledged
MccController, 266
callbackBooking
MccController, 254
Scheduler, 305
callbackExperimentUploadFailure
Controller, 177
callbackFailed
MccController, 266
callbackList
MccController, 254
callbackListOfOpportunities
Controller, 177
Scheduler, 306
callbackListOpportunities
Controller, 193
callbackMCS
Scheduler, 306
callbackNewData
MccController, 266
Scheduler, 306
callbackReceivedData
Controller, 177
callbackTimeoutMCC
Scheduler, 307
callbackTimeoutMCS
Scheduler, 307
cancelBooking
MccClient, 251
MccController, 255
cancelExternalBookingsOfExperiment
Scheduler, 307
Cancelled
defs.h, 343
cancelled, 158
cancelled, 159
canceler, 159
canceler, 159
sat, 159
Ch
sha2, 315, 316
challenge
Controller::SessionInfo, 196
ServerConnectivity, 74
Controller::ChallengeRequest
ServerConnectivity, 67
changePassword, 159
changePassword, 160
Controller, 177
canceler, 160
execute, 160
ServerConnectivity, 67
checkAccountIsRegistered
DatabaseAbstraction, 215
checkAllSatellites
MccController, 259
Scheduler, 305
checkForTimeouts
MccController, 255
McsController, 267
checkResult
DatabaseAbstraction, 216
checkSatellite
MccController, 259
McsController, 267
Scheduler, 305
cleanExpiredSessions
Controller, 178
clearQueue
Queue, 285
clearSelection
Commands, 12
clearer
MccInfo, 257
McsInfo, 272
client/ Directory Reference, 9
client/commands.cpp, 79
client/commands.h, 80
client/commandseditdialog.cpp, 80
client/commandseditdialog.h, 81
client/communicationlogsdialog.cpp, 81
client/communicationlogsdialog.h, 81
client/comntest.cpp, 82
client/createexperimentdialog.cpp, 82
client/createexperimentdialog.h, 83
client/experimentdialog.cpp, 84
client/experimentdialog.h, 84
client/logindialog.cpp, 85
client/logindialog.h, 86
client/main.cpp, 87
client/mainwindow.cpp, 87
client/mainwindow.h, 89
client/manageobservers.cpp, 89
client/manageobservers.h, 89
client/managepayloadsdialog.cpp, 90
client/managepayloadsdialog.h, 90
client/mangesatellitesdialog.cpp, 91
client/mangesatellitesdialog.h, 91
client/manageusersdialog.cpp, 91
client/manageusersdialog.h, 92
client/multihttp.cpp, 92
client/multihttp.h, 93
closeOtherWindows
MainWindow, 36
closing
MainWindow, 38
Command, 162
~Command, 164
Command, 164
commandParam, 167
getID, 165
getName, 165
getParameters, 165
getPayloadID, 165
identifier, 167
instantiate, 166
name, 167
needs_storage, 167
needs_update, 167
payload, 167
setName, 166
setParameters, 166
setPayloadID, 166
setUpdate, 167
store, 167
Command::CommandException, 168
Command::CommandException, 168
CommandID
defs.h, 342
defaultValue, 169
defaultValue, 169
defaultValue, 169
maxValue, 169
operator==, 169
minValue, 169
operator==, 169
paramType, 169
commandParam
Command, 167
Commands, 10

INDEX

client/commands.cpp, 79
client/commands.h, 80
client/commandseditdialog.cpp, 80
client/commandseditdialog.h, 81
client/communicationlogsdialog.cpp, 81
client/communicationlogsdialog.h, 81
client/comntest.cpp, 82
client/createexperimentdialog.cpp, 82
client/createexperimentdialog.h, 83
client/experimentdialog.cpp, 84
client/experimentdialog.h, 84
client/logindialog.cpp, 85
client/logindialog.h, 86
client/main.cpp, 87
client/mainwindow.cpp, 87
client/mainwindow.h, 89
client/manageobservers.cpp, 89
client/manageobservers.h, 89
client/managepayloadsdialog.cpp, 90
client/managepayloadsdialog.h, 90
client/mangesatellitesdialog.cpp, 91
client/mangesatellitesdialog.h, 91
client/manageusersdialog.cpp, 91
client/manageusersdialog.h, 92
client/multihttp.cpp, 92
client/multihttp.h, 93
closeOtherWindows
MainWindow, 36
closing
MainWindow, 38
Command, 162
~Command, 164
Command, 164
commandParam, 167
getID, 165
getName, 165
getParameters, 165
getPayloadID, 165
identifier, 167
instantiate, 166
name, 167
needs_storage, 167
needs_update, 167
payload, 167
setName, 166
setParameters, 166
setPayloadID, 166
setUpdate, 167
store, 167
Command::CommandException, 168
Command::CommandException, 168
CommandID
defs.h, 342
defaultValue, 169
defaultValue, 169
defaultValue, 169
maxValue, 169
operator==, 169
minValue, 169
operator==, 169
paramType, 169
commandParam
Command, 167
Commands, 10

DDD 0.1 433
addCommand, 12
addParameter, 12
clearSelection, 12
commandList::currentChanged, 13
commandModel, 14
Commands, 12
getCommand, 13
getCommandName, 13
IdRole, 12
ParamDefaultRole, 11
ParamMaxRole, 12
ParamMinRole, 11
selectCommand, 13
selectionChanged, 14
setupCommands, 14
setupParameterHeader, 14
setupParameters, 14
Controller::CallbackHandleOpportunities, 194
Component
SingleLog, 319
clearSelection, 12
controller
Controller::ThreadingInformation, 198
connect
DatabaseAbstraction, 216
connection
ConnectionManager, 170
DatabaseAbstraction, 220
ConnectionManager, 169
connection, 170
getCommand, 13
getConnect, 170
conntest.cpp
main, 82
Controller, 170
Controller::CallbackHandleOpportunities, 194
addAccountToPayload, 176
addCommand, 176
callbackExperimentUploadFailure, 177
callbackListOpportunities, 177
callbackListOpportunities, 193
callbackReceivedData, 177
changePassword, 177
cleanExpiredSessions, 178
Controller, 176
createAccount, 178
createExperiment, 178
createPayload, 179
createSatellite, 179
deleteExperiment, 179
getAccountFromSession, 180
getChallenge, 180
getCommand, 180
getAddress, 181
getAddress, 182
getAddress, 183
getAddress, 184
getChallenge, 180
getCommand, 180
login, 185
LoginInfo, 176
markForDeletion, 186
markForHighResolutionDownload, 186
INDEX

received Thumbnails, 193
register Account, 186
register Payload, 187
register Satellite, 187
remove Account, 187
remove Account From Payload, 187
remove Command, 188
remove Payload, 188
remove Satellite, 188
request Thumbnail, 189
Satellite Info, 176
schedule Experiment, 189
scheduler, 193
send Command, 189
sessions, 193
Single Log, 319
unregister Account, 190
unregister Payload, 190
unregister Satellite, 190
unschedule Experiment, 190
update Account, 191
update Command, 191
update Experiment, 191
update Keepalive, 192
update Payload, 192
update Satellite, 192
wait Callback Data, 193
wait Callback Opportunities, 193
controller
acknowledged, 126
booked, 149
cancelled, 159
change Password, 160
Client Connectivity, 162
Controller::Thread Argument Int, 197
Controller::Thread Argument String Int, 197
create Experiment, 199
failed, 234
get Challenge And Session, 235
get Experiment, 237
get Experiments, 238
get List Of Opportunities, 239
get List Of Thumbnails, 240
get Payload, 241
get Satellite, 243
get Thumbnail, 244
list Failure, 246
list Of Opportunities, 246
login, 249
logout, 250
Mcc Listener, 259
Mes Listener, 274
new Data, 275
request Failure, 291
request Rejected, 292
schedule Experiment, 302
Scheduler, 313
unschedule Experiment, 322
update Experiment, 323
controller.cpp
await List Of Opportunities, 374
await New Data, 374
KEEPALIVE_TIME, 374
safety Thread, 374
Controller::Callback Handle Opportunities, 193
bookings, 194
completed, 194
id, 194
status, 194
Controller::Controller Exception, 194
ControllerException, 194
Controller::Experiment Info, 195
identifier, 195
name, 195
payload, 195
status, 195
Controller::Payload Info, 195
identifier, 196
name, 196
satellite ID, 196
Controller::Session Info, 196
account ID, 196
challenge, 196
last Keep alive, 196
last Retrieved Thumbnail, 196
Controller::Thread Argument Int, 197
controller, 197
id, 197
Controller::Thread Argument String Int, 197
controller, 197
id, 197
session, 197
Controller::Threading Information, 197
condition, 198
mutex, 198
ControllerException
Controller::Controller Exception, 194
create Account
Controller, 178
create Booking
Scheduler, 308
Created
defs.h, 343
create Experiment, 198
Controller, 178
controller, 199
create Experiment, 199
execute, 199

DDD 0.1 435
SetDataItemLength, 206
setExperimentID, 206
setGatheredTime, 206
setMarkedForDeletion, 206
setName, 207
setThumbnail, 207
setThumbnailLength, 207
setType, 207
setUpdate, 208
store, 208
thumbnail, 209
thumbnailLength, 209
type, 209
data
Thumbnail, 76
data.cpp
getAllDataIDs, 336
data.h
getAllDataIDs, 338
Data::DataException, 209
DataException, 209
DatabaseAbstraction, 210
~DatabaseAbstraction, 215
accountAddExperiment, 215
accountAddPayload, 215
accountExists, 215
accountHasRightsToExperiment, 215
accountHasRightsToPayload, 215
accountHasRightsToSatellite, 215
accountRemoveExperiment, 215
accountRemovePayload, 215
addLog, 215
checkAccountIsRegistered, 215
checkResult, 216
connect, 216
connection, 220
DatabaseAbstraction, 215
db, 220
executeAndProcessQuery, 216
executeQuery, 216
getAllBookingIDs, 216
getAllCommandIDs, 216
getAllDataIDs, 216
getAllExperimentIDs, 216
getAllPayloadIDs, 216
getAllSatelliteIDs, 216
getAvailableSatellites, 216
dataList, 216
getDataList, 216
getExperimentAtTime, 216
getExternalBookingsOfExperiment, 216
getFirstBooking, 217
getInternalBooking, 217
getPayloadExperiments, 217
getPayloadListOfCommands, 217
INDEX

getQueue, 217
getThumbnailList, 217
g getTimeBookingIDs, 217
host, 220
newQuery, 217
numberOfRetries, 220
passwd, 220
removeExperiment, 217
removeQueue, 217
removeQueueItem, 218
retrieveAccount, 218
retrieveBooking, 218
retrieveCommand, 218
retrieveData, 218
retrieveExperiment, 218
retrieveLog, 218
retrieveLogs, 218
retrievePayload, 218
removeQueueItem, 219
retrieveSatellite, 219
storeQuery, 219
updateAccount, 219
updateBooking, 219
updateCommand, 219
updateData, 219
updateExperiment, 219
updatePayload, 220
updateQueueItem, 220
updateSatellite, 220
user, 220
databaseabstraction.cpp
INITIAL_SLEEP_TIME, 377
DatabaseAbstraction::DatabaseException, 220
DatabaseException, 221
DatabaseException::DatabaseException, 221
DataException, 209
Data::DataException, 209
dataList, 342
dataItem
Data, 208
dataItemLength
Data, 208
DataList
defs.h, 342
dataParam
DataModel
ObservationDataDialog, 47
db
DatabaseAbstraction, 220
defaultValue
CommandParam, 169
Defined
defs.h, 343

Accept, 343
AccountID, 342
AccountType, 343
Action, 343
Administrator, 343
BookingID, 342
BookingList, 342
BookingStatus, 343
BookingType, 343
Boolean, 344
Cancelled, 343
CommandID, 342
Created, 343
DataID, 342
DataList, 342
Defined, 343
Direction, 344
Double, 344
Erase, 343
Exec, 343
Executed, 344
Executing, 344
ExperimentID, 342
ExperimentParts, 342
ExperimentStatus, 344
External, 344
Failure, 343
Fetch, 343
Finished, 343
GSSID, 342
Integer, 344
Internal, 344
ItemID, 342
Kill, 343
List, 343
ListOfCommands, 342
ListOfExperimentCommands, 342
Local, 344
LogID, 342
Observer, 343
OneWay, 344
ParamType, 344
PayloadID, 343
Pending, 343
Rejected, 343
Run, 343
Running, 343
SatelliteID, 343
Schedule, 343
Scheduled, 344
Scientist, 343
Store, 343
TwoWay, 344
getPayloadID, 226
getSchedule, 226
getStatus, 227
getThumbnailList, 227
id, 21
identifier, 231
instantiate, 227
listOfCommands, 231
modifyCommand, 227
name, 21, 231
needs_storage, 231
needs_update, 231
payload, 231
possibleCommands, 21
removeCommand, 228
removeExperiment, 228
schedule, 228
scheduledFrom, 21, 231
scheduledTo, 21, 231
setListOfCommands, 229
setName, 229
setPayloadID, 229
setScheduledTo, 230
setStatus, 230
setUpdate, 230
status, 231
store, 230
timedCommands, 21
unschedule, 230
experiment
Data, 208
experiment.cpp
getAllExperimentIDs, 345
experiment.h
getAllExperimentIDs, 346
experiment1
SelectObserveDialog, 61
experiment1Id
Observe, 52
experiment2
SelectObserveDialog, 61
experiment2Id
Observe, 52
Experiment::ExperimentException, 231
ExperimentException, 232
ExperimentCommand, 232
command, 233
currentValue, 233
operator==, 233
paramType, 233
time, 233
experimentCreated
CreateExperimentDialog, 19
ServerConnectivity, 68
ExperimentDialog, 21
~ExperimentDialog, 25
addTimedCommand, 25
closeEvent, 26
CommandIdRole, 25
currentTimedCommandChanged, 26
ExperimentDialog, 25
experimentId, 29
experimentName, 29
experimentReceived, 26
experimentScheduled, 26
experimentUnscheduled, 26
GssId, 25
GssIdRole, 24
Interval, 25
IntervalFromRole, 24
IntervalToRole, 24
ItemType, 25
on_pushButtonBox_clicked, 26
on_commandAddButton_clicked, 27
on_commandDeleteButton_clicked, 27
on_commandEditButton_clicked, 27
on_commands_selectionChanged, 27
on_editButton_clicked, 27
on_observersButton_clicked, 27
on_scheduleButton_clicked, 28
on_scheduleDateEdit_dateChanged, 28
on_scheduleExperimentButton_clicked, 28
on_unscheduleButton_clicked, 28
opportunitiesReceived, 28
passesModel, 29
payloadId, 29
satelliteId, 29
scheduledFrom, 29
sequenceModel, 29
setExperimentChanged, 28
setupModels, 28
switchSchedule, 29
TypeRole, 24
ui, 29
ValueRole, 25
ExperimentException
Experiment::ExperimentException, 232
ExperimentID
defs.h, 342
experimentID
Booking, 156
SingleLog, 319
experimentId
CreateExperimentDialog, 20
ExperimentDialog, 29
Thumbnail, 76
experimentInfoToXML
AdminLowlevel, 136  
DatabaseAbstraction, 216  
Scheduler, 309  

getBookings  
Scheduler, 310  

getchallenge  
Controller, 180  
getchallengeAndSession, 234  
controller, 235  
execute, 235

getchallengeAndSession, 235  

getcode  
Commands, 13  
Controller, 180  

getcodeName  
Commands, 13  

getcodeConnection  
ConnectionManager, 170  
getcodeItem  
Data, 203  
getcodeItemLength  
Data, 203  
getcodeList  
AdminLowlevel, 136  
DatabaseAbstraction, 216  
Experiment, 225  
getcodeData  
Controller, 181  
getcodeDirection  
Booking, 153  

getcodeEmail  
Account, 118  
getcodeEndTime  
Booking, 153  
getcodeEnumType  
sha2, 316  
getcodeExperiment  
Controller, 181  
controller, 237  
execute, 236  
getcodeExperiment  
Satellite, 295  

getcodeExperimentAtTime  
AdminLowlevel, 136  
DatabaseAbstraction, 216  

getcodeExploremDetails  
ServerConnectivity, 68  
getcodeExploremID  
Booking, 153  
Data, 203  

getcodeExploremIDs  
Controller, 181  
getcodeExploremListModel  
ServerConnectivity, 68  

getExperiments, 237  
controller, 238  
execute, 238  
getcodeExplorem, 238  
Payload, 279  

getcodeExternalBookingsOfExplorem  
AdminLowlevel, 136  
booking.cpp, 365  
booking.h, 367  
DatabaseAbstraction, 216  

getcodeFirstBookings  
AdminLowlevel, 137  
booking.cpp, 365  
booking.h, 368  
DatabaseAbstraction, 217  

getcodeFullName  
Account, 119  
getcodeGatheredTime  
Data, 203  
getcodeGssID  
Booking, 153  
GetHash  
sha2, 316  
getcodeHistory  
Queue, 286  

getcodeHostname  
MccController, 255  
McsController, 268  
XmlRpcController, 327  
getcodeID  

Account, 119  
Booking, 153  
Command, 165  
Data, 204  
Experiment, 225  

Payload, 279  
Satellite, 296  
getcodeId  
CreateExperimentDialog, 19  

getcodeInternalBookings  
AdminLowlevel, 137  
booking.cpp, 366  
booking.h, 368  
DatabaseAbstraction, 217  

getcodeLastReceivedThumbnail  
Controller, 182  

getcodeLastRetrievedThumbnail  
Controller, 182  

getcodeListOfCommands  
Experiment, 225  

Payload, 279  

getcodeListOfOpportunities  
Controller, 182  
controller, 239  

DDD 0.1
INDEX

Data, 205
GetTypeString
    sha2, 316
11390  getUniqueListOfExperiment
Scheduler, 310
GSSID
defs.h, 342
GssId
11395  ExperimentDialog, 25
gssID
Booking, 156
gssid
BookingInfo, 158
GssIdRole
    ExperimentDialog, 24
    hash
    ServerConnectivity, 68
    utils.h, 364
11405  hashedNewPassword
ServerConnectivity, 75
hashedPassword
    ServerConnectivity, 68
11410  hashedPassword_
ServerConnectivity, 75
hasHighResolution
Controller, 184
hasRightsToExperiment
    Account, 120
11415  hasRightsToPayload
    Account, 120
hasRightsToSatellite
    Account, 120
11420  HexHash
    sha2, 316
history
    QueueItem, 289
historySize
    Queue, 286
11425  host
    DatabaseAbstraction, 220
https
    MultiHttp, 45
11430  HttpSlot
    ServerConnectivity, 66
i
    ParamTypes, 276
id
11435  Controller::CallbackHandleOpportunities, 194
Controller::ThreadArgumentInt, 197
Controller::ThreadArgumentStringInt, 197
Experiment, 21
PossibleCommand, 53
Thumbnail, 76
identifier
    Account, 124
    Booking, 156
    Command, 167
11440  Controller::ExperimentInfo, 195
Controller::PayloadInfo, 196
Data, 208
Experiment, 231
Payload, 282
Satellite, 299
IdRole
    Commands, 12
ServerConnectivity, 66
IMSETY_VERSION
    version.h, 105, 405
infoList
    MccController, 257
    McsController, 271
Init
11450  sha2, 316
    INITIAL_SLEEP_TIME
    databaseabstraction.cpp, 377
initializing
    CreateExperimentDialog, 20
11455  ObservationDataDialog, 47
SelectObserveDialog, 61
instance
    ServerConnectivity, 69
instance,
11460  ServerConnectivity, 75
instanti ate
    Account, 121
    Booking, 154
    Command, 166
11465  Data, 205
    Experiment, 227
    Payload, 280
    Queue, 286
    Satellite, 296
11470  Integer
    defs.h, 344
Internal
    defs.h, 344
internal
11475  queues, 290
interpretBooleanResponse
ServerConnectivity, 69
interpretChallengeResponse
ServerConnectivity, 69
interpretIntResponse
ServerConnectivity, 69
interpretListOfOpportunities
ServerConnectivity, 69
log.h
  addLog, 352
  log, 353
  retrieveLog, 353
  retrieveLogs, 354
LogException, 247
  LogException, 247
loggedIn
  ServerConnectivity, 75
LogID
defs.h, 342
login, 247
  Controller, 185
  controller, 249
  execute, 248
  login, 248
ServerConnectivity, 70
loginAniTimeLine
  LoginDialog, 33
LoginDialog, 29
  ~LoginDialog, 31
  closeEvent, 31
loginAniTimeLine, 33
LoginDialog, 31
  noPasswordChange, 33
  on_cancelButton_clicked, 31
  on_loginButton_clicked, 32
  on_preferencesButton_clicked, 32
  reject, 32
  requestFinished, 32
  setAniFrame, 32
ui, 33
loginFinished
  ServerConnectivity, 70
LoginInfo
  Controller, 176
LoginRequest
  ServerConnectivity, 67
logout, 249
  Controller, 185
  controller, 250
  execute, 250
  logout, 250
ServerConnectivity, 70
logoutFinished
  ServerConnectivity, 70
LogoutRequest
  ServerConnectivity, 67
logVisible
  ObservationView, 49, 50
m_boolEnded
  sha2, 317
m_boolIsBigEndian
  sha2, 317
m_chrHexHash
  sha2, 318
m_chrRawHash
  sha2, 318
m_digest
  sha2, 318
m_strHash
  sha2, 318
m_Type
  sha2, 318
Main
  main
    conntest.cpp, 82
    main.cpp, 87, 380
MainWindow, 33
  ~MainWindow, 36
  closeEvent, 36
  closeOtherWindows, 36
  closing, 38
  dontLogout, 38
  experimentListUpdated, 37
  getSelectedExperiment, 37
  getSelectedExperiments, 37
  MainWindow, 36
  on_actionAbout_triggered, 37
  on_actionAboutQt_triggered, 37
  on_actionCommunicationLogs_triggered, 37
  on_actionEditExperiment_triggered, 37
  on_actionLogout_triggered, 37
  on_actionManageObservers_triggered, 37
  on_actionManageSatellites_triggered, 37
  on_actionManageUsers_triggered, 37
  on_actionNewExperiment_triggered, 37
  on_actionObservationData_triggered, 37
  on_actionObserveExperiment_triggered, 37
  on_actionPreferences_triggered, 37
  on_actionRefreshList_triggered, 37
  on_actionScheduleExperiment_triggered, 37
  on_experimentsTreeView_doubleClicked, 37
  openAdmin, 38
  selectionChanged, 38
  setupModels, 38
  ui, 38
Maj
  sha2, 316
makeHttpHeader
ServerConnectivity, 70
ManageObservers, 38
  ~ManageObservers, 39
  addUser, 39
  ManageObservers, 39
observerModel, 40

DDD 0.1 445
setupModels, 39
ui, 40
userModel, 40
ManagePayloadsDialog, 40
  ~ManagePayloadsDialog, 40
ManagePayloadsDialog, 40
  on_addButton_clicked, 41
  on_editButton_clicked, 41
  ui, 41
ManageSatellitesDialog, 41
  ~ManageSatellitesDialog, 42
ManageSatellitesDialog, 42
  on_addButton_clicked, 42
  on_editButton_clicked, 42
  ui, 42
ManageUsersDialog, 42
  ~ManageUsersDialog, 43
ManageUsersDialog, 43
  on_addButton_clicked, 43
  on_editButton_clicked, 43
  ui, 43
markedForDeletion
  Data, 208
markForDeletion
Controller, 186
markForHighResolutionDownload
  Controller, 186
maxValue
  CommandParam, 169
MCC
  SingleLog, 319
MccClient, 250
cancelBooking, 251
getListOfopportunities, 251
MccClient, 251
myCarriageParmPtr, 252
myClientPtr, 252
myUrl, 252
requestBooking, 252
MccController, 252
  bookingIds, 257
callbackBooking, 254
callbackList, 254
cancelBooking, 255
checkForTimeouts, 255
checkSatellite, 255
getHostname, 255
getListOfopportunities, 256
getNewSocket, 256
infoList, 257
lastUsed, 257
MccController, 254
removeBooking, 256
requestBooking, 256
11790  scheduler, 257
timeouts, 257
mccController
  Scheduler, 313
mccController.cpp
  runMccTimeouts, 382
  SLEEP, 382
  TIMEOUT_RESPONSE, 382
mccGeneral.cpp
  bookingInfoFromClass, 384
  bookingInfoToClass, 384
  bookingListFromXML, 385
  intervalFromClass, 385
mccGeneral.h
  bookingInfoFromClass, 386
  bookingInfoToClass, 386
  bookingListFromXML, 387
  intervalFromClass, 387
MccInfo, 257
caller, 257
thread, 257
MccListener, 258
  ~MccListener, 259
controller, 259
MccListener, 259
  myAbyssServerPtr, 259
run, 259
sat, 259
MccSlot
  ServerConnectivity, 66
mccUrl
  Satellite, 299
MCS
  SingleLog, 319
McsClient, 259
erase, 261
exec, 261
fetch, 261
kill, 262
list, 262
McsClient, 261
myCarriageParmPtr, 263
myClientPtr, 263
myUrl, 263
run, 262
schedule, 262
store, 263
McsController, 263
  bookingIds, 271
callbackAcknowledged, 266
callbackFailed, 266
callbackNewData, 266
checkAllSatellites, 267
checkForTimeouts, 267
checkSatellite, 267
erase, 267
eexec, 268
fetch, 268
getHostname, 268
getNewSocket, 269
infoList, 271
kill, 269
lastUsed, 271
list, 269
McsController, 266
removeBooking, 269
run, 270
schedule, 270
scheduler, 271
store, 270
timeouts, 272
unschedule, 271
mcsController
Scheduler, 313
mcsController.cpp
runMcsTimeouts, 392
SLEEP, 392
TIMEOUT_RESPONSE, 392
mcsGeneral.cpp
experimentInfoToXML, 394
experimentPartToXML, 394
mcsGeneral.h
experimentInfoToXML, 395
experimentPartToXML, 395
McsInfo, 272
client, 272
port, 272
thread, 272
McsListener, 272
~McsListener, 274
controller, 274
McsListener, 273
myAbyssServerPtr, 274
run, 274
sat, 274
mcsUrl
Satellite, 299
MEMCPY_BCOPY
sha2, 316
MEMSET_BZERO
sha2, 316
message
SingleLog, 319
minPasswordLength
Preferences, 55
minValue
CommandParam, 169
modifyCommand
Experiment, 227
MultiHttp, 43
~MultiHttp, 44
abort, 44
https, 45
MultiHttp, 44
request, 44
requestFinished, 45
requestFinishedSlot, 45
requestPending, 45
requestTypes, 45
responses, 45
setHost, 45
setProxy, 45
multiHttp
ServerConnectivity, 75
mutex
Controller::ThreadingInformation, 198
myAbyssServerPtr
ClientConnectivity, 162
MccListener, 259
McsListener, 274
XmlRpcListener, 329
myCarriageParmPtr
MccClient, 252
McsClient, 263
XmlRpcClient, 325
myClientPtr
MccClient, 252
McsClient, 263
XmlRpcClient, 325
myUrl
MccClient, 252
McsClient, 263
XmlRpcClient, 325
name
Command, 167
Controller::ExperimentInfo, 195
Controller::PayloadInfo, 196
Data, 208
Experiment, 21, 231
Payload, 282
PossibleCommand, 53
Satellite, 299
Thumbnail, 76
needs_storage
Account, 124
Command, 167
Data, 208
Experiment, 231
Payload, 282
Satellite, 299
needs_update
  Account, 124
  Command, 167
  Data, 209
11925  Experiment, 231
  Payload, 282
  Satellite, 299
newData, 274
  controller, 275
  execute, 275
  newData, 275
  sat, 275
newObservation
  ObservationView, 49
11930  newQuery
  DatabaseAbstraction, 217
  newSingleThumbnail
  ServerConnectivity, 70
  newThumbnail
11940  ObservationDataDialog, 47
  Observe, 52
  ServerConnectivity, 70
noPasswordChange
  LoginDialog, 33
11950  numberOfRetries
  DatabaseAbstraction, 220
ObservationDataDialog, 45
  ~ObservationDataDialog, 47
  dataModel, 47
  initializing, 47
  newThumbnail, 47
  ObservationDataDialog, 47
  onFilterButton_clicked, 47
  on_payloadComboBox_currentIndexChanged, 47
11965  on_satelliteComboBox_currentIndexChanged, 47
  thumbnailSize, 47
  ui, 47
ObservationView, 47
  ~ObservationView, 49
  appendLog, 49
  Left, 49
  logVisible, 49, 50
  newObservation, 49
ObservationView, 49
  on_addButton_clicked, 49
  Orientation, 49
  orientation, 50
postProcVisible, 50
Right, 49
setLogVisible, 50
setTitle, 50
ui, 50
Observe, 50
  ~Observe, 52
  closeEvent, 52
11980  experiment1Id, 52
  experiment2Id, 52
  newThumbnail, 52
  Observe, 51
  ui, 52
Observer
defs.h, 343
observerModel
  ManageObservers, 40
ObserveSlot
  ServerConnectivity, 66
  on_actionAbout_triggered
  MainWindow, 37
11990  on_actionAboutQt_triggered
  MainWindow, 37
  on_actionCommunicationLogs_triggered
  MainWindow, 37
  on_actionEditExperiment_triggered
  MainWindow, 37
  on_actionLogout_triggered
  MainWindow, 37
  on_actionManageObservers_triggered
  MainWindow, 37
  on_actionManageSatellites_triggered
  MainWindow, 37
12000  on_actionManageUsers_triggered
  MainWindow, 37
  on_actionNewExperiment_triggered
  MainWindow, 37
  on_actionObserveExperiment_triggered
  MainWindow, 37
  on_actionPreferences_triggered
  MainWindow, 37
12015  on_actionRefreshList_triggered
  MainWindow, 37
  on_actionScheduleExperiment_triggered
  MainWindow, 37
  on_addButton_clicked
  ManagePayloadsDialog, 41
  ManageSatellitesDialog, 42
  ManageUsersDialog, 43
ObservationView, 49
  UserSelectionWidget, 79
12025  on_buttonBox_accepted
  CreateExperimentDialog, 20
Preferences, 55

448 DDD 0.1
INDEX

on_pushButtonBox_clicked
    ExperimentDialog, 26
    Preferences, 55
on_pushButtonBox_rejected
    Preferences, 55
on_pushButtonCancel_clicked
    LoginDialog, 31
on_changePasswordPushButton_clicked
    Preferences, 55
on_commandAddButton_clicked
    ExperimentDialog, 27
on_commandDeleteButton_clicked
    ExperimentDialog, 27
on_commandEditButton_clicked
    ExperimentDialog, 27
on_commands_selectionChanged
    ExperimentDialog, 27
on_editButton_clicked
    ExperimentDialog, 27
    ManagePayloadsDialog, 41
    ManageSatellitesDialog, 42
    ManageUsersDialog, 43
on_experiment1GroupBox_toggled
    SelectObserveDialog, 61
on_experiment2GroupBox_toggled
    SelectObserveDialog, 61
on_experimentsTreeView_doubleClicked
    MainWindow, 37
on_filterPushButton_clicked
    ObservationDataDialog, 47
on_pushButtonLogin_clicked
    LoginDialog, 32
on_newPasswordEdit_textChanged
    Preferences, 55
on_observersPushButton_clicked
    ExperimentDialog, 27
on_payload1ComboBox_currentIndexChanged
    SelectObserveDialog, 61
on_payload2ComboBox_currentIndexChanged
    SelectObserveDialog, 61
on_payloadComboBox_currentIndexChanged
    ObservationDataDialog, 47
on_preferencesPushButton_clicked
    LoginDialog, 32
on_pushButtonRemove_clicked
    UserSelectionWidget, 79
on_retypedNewPasswordEdit_textChanged
    Preferences, 55
on_satellite1ComboBox_currentIndexChanged
    SelectObserveDialog, 61
on_satellite2ComboBox_currentIndexChanged
    SelectObserveDialog, 61
on_satelliteComboBox_currentIndexChanged
    CreateExperimentDialog, 20
ObservationDataDialog, 47
on_pushButtonSchedule_clicked
    ExperimentDialog, 28
on_pushButtonScheduleDateEdit_dateChanged
    ExperimentDialog, 28
on_pushButtonScheduleExperimentButton_clicked
    ExperimentDialog, 28
on_pushButtonSearchUserEdit_textChanged
    UserSelectionWidget, 79
on_pushButton Unschedule_clicked
    ExperimentDialog, 28
OneWay
defs.h, 344
openAdmin
MainWindow, 38
operator=
    Data, 205
operator==
    BookingInfo, 158
    CommandParam, 169
    ExperimentCommand, 233
    Interval, 244
    QueueItem, 289
opportunitiesReceived
    ExperimentDialog, 28
    ServerConnectivity, 70
Opportunity, 52
    end, 53
    gssid, 53
    remarks, 53
    start, 53
    twoway, 53
Orientation
    ObservationView, 49
    orientation
    ObservationView, 50
paramDefault
    PossibleCommand, 53
ParamDefaultRole
    Commands, 11
    Commands, 14
paramMax
    PossibleCommand, 53
    Commands, 12
    Commands, 11
ParamMinMax
    PossibleCommand, 53
ParamMinMaxRole
    Commands, 11
ParamType
defs.h, 344
paramType
INDEX

processExperiment
    ServerConnectivity, 70
processExperimentsList
    ServerConnectivity, 70
processItem
    Scheduler, 311
processPayload
    ServerConnectivity, 70
processPossibleCommand
    ServerConnectivity, 70
processQueue
    Scheduler, 311
processSatellite
    Scheduler, 312
    ServerConnectivity, 71
processTimedCommand
    ServerConnectivity, 71
proxyHost
    ServerConnectivity, 71
proxyHost_
    ServerConnectivity, 75
proxyPort
    ServerConnectivity, 71
proxyPort_
    ServerConnectivity, 75
pushQueueItem
    Queue, 287
QDialog, 56
QMainWindow, 56
QObject, 57
QStandardItemModel, 59
Queue, 283
~Queue, 285
clearQueue, 285
deleteHistoryItem, 285
deleteQueueItem, 285
empty, 286
getHistory, 286
getcheck, 286
historySize, 286
instantiate, 286
popQueueItem, 287
pushQueueItem, 287
Queue, 284
queueList, 288
queueSize, 287
retrievelHistoryItem, 287
running, 288
satelliteID, 288
type, 288
Queue::QueueException, 288
QueueItem, 289
action, 289
history, 289
intVal, 289
itemID, 289
operator==, 289
payloadID, 290
satID, 290
strVal, 290
type, 290
queueList
    Queue, 288
    Scheduler, 313
Queues, 290
    external, 290
    internal, 290
    Queues, 290
queueSize
    Queue, 287
QWidget, 59
RawHash
    sha2, 316
readBoolean
    ServerConnectivity, 71
readByteArray
    ServerConnectivity, 71
readInt
    ServerConnectivity, 71
readString
    ServerConnectivity, 71
readVariant
    ServerConnectivity, 72
receivedThumbnails
    registerAccount
    Account, 121
    Controller, 186
    registered
    Account, 124
    Payload, 282
    Satellite, 299
    registerPayload
    Controller, 187
    Payload, 280
    registerSatellite
    Controller, 187
    Satellite, 297
    reject
    LoginDialog, 32
    Rejected
defs.h, 343
remarks

DDD 0.1 451
seed_rand, 400
ROTL32
sha2, 316
ROTR32
sha2, 316
ROTR64
sha2, 316
Run
defs.h, 343
run
ClientConnectivity, 162
MccListener, 259
McsClient, 262
McsController, 270
McsListener, 274
Scheduler, 312
XmlRpcListener, 329
runListener
xmlRpcController.cpp, 406
xmlRpcController.h, 408
runMccTimeouts
mccController.cpp, 382
runMcsTimeouts
mcsController.cpp, 392
Running
defs.h, 343
running
Queue, 288
runSatellite
scheduler.cpp, 402
scheduler.h, 404
runScheduler
scheduler.cpp, 402
scheduler.h, 404
runUpdateThread
scheduler.cpp, 402
safetyThread
controller.cpp, 374
Sat
acknowledged, 126
booked, 149
cancelled, 159
failed, 234
listFailure, 246
listOfOpportunities, 246
MccListener, 259
McsListener, 274
newData, 275
requestFailure, 291
requestRejected, 292
SchedInfo, 300
Satellite, 293
~Satellite, 295
getExperiment, 295
getID, 296
getMccUrl, 296
getMcsUrl, 296
getName, 296
identifier, 299
isRegistered, 297
mccUrl, 299
mcsUrl, 299
name, 299
needs_storage, 299
needs_update, 299
registered, 299
registerSatellite, 297
Satellite, 204, 205
setMccUrl, 297
setMcsUrl, 297
setName, 298
setUpdate, 298
store, 298
unregisterSatellite, 298
satellite.cpp
getAllSatelliteIDs, 357
satellite.h
getAllSatelliteIDs, 359
Satellite::SatelliteException, 299
SatelliteException, 300
SatelliteException
Satellite::SatelliteException, 300
SatelliteID
defs.h, 343
satelliteID
Controller::PayloadInfo, 196
Payload, 282
Queue, 288
SingleLog, 319
satelliteId
ExperimentDialog, 29
SatelliteInfo
Controller, 176
SatelliteItem
ServerConnectivity, 67
satelliteToXMLStruct
xmlRpcUtils.cpp, 413
xmlRpcUtils.h, 417
satID
QueueItem, 290
SchedInfo, 300
sat, 300
scheduler, 300
type, 300
Schedule
defs.h, 343
sigma 512, 317
StringHash, 317
Update, 317
sha2.cpp
K1_0_TO_19, 359
K1_20_TO_39, 359
K1_40_TO_59, 359
K1_60_TO_79, 360
K256, 360
sha1_initial_hash_value, 360
sha224_initial_hash_value, 360
sha256_initial_hash_value, 360
sha384_initial_hash_value, 360
sha512_initial_hash_value, 361
sha_hex_digits, 361
sha2.h
SHA1_DIGESTC_LENGTH, 362
SHA1_DIGESTC_STRING_LENGTH, 362
SHA224_DIGESTC_LENGTH, 362
SHA224_DIGESTC_STRING_LENGTH, 362
SHA256_DIGESTC_LENGTH, 362
SHA256_DIGESTC_STRING_LENGTH, 362
SHA384_DIGESTC_LENGTH, 363
SHA384_DIGESTC_STRING_LENGTH, 363
SHA512_DIGESTC_LENGTH, 363
SHA512_DIGESTC_STRING_LENGTH, 363
Sigma0_256
Sigma0_512
Sigma1_256
Sigma1_512
SimpleRequest
ServerConnectivity
fullsetModel, 79
fullsetProxyModel, 79
on_addButton_clicked, 79
on_removeButton_clicked, 79
on_searchUserEdit_textChanged, 79
setFullsetModel, 79
setSubsetModel, 79
subsetModel, 79
ui, 79
UserSelectionWidget, 78
usertype
ServerConnectivity, 74
usertype
ServerConnectivity, 75
utils.h
hash, 364
toString, 364
ValueRole
ExperimentDialog, 25
version.h
IMSETY_VERSION, 105, 405

waitCallbackData
Controller, 193
waitCallbackOpportunities
Controller, 193

xmlRpc_c::method, 323
XmlRpcClient, 324

~XmlRpcClient, 325
myCarriageParmPtr, 325
myClientPtr, 325
myUrl, 325
XmlRpcClient, 324, 325

XmlRpcController, 325
~XmlRpcController, 327
bookingIds, 327
getHostname, 327
getNewSocket, 327
lastUsed, 327
removeBooking, 327
scheduler, 327
timeouts, 327
xmlRpcController.cpp
runListener, 406
xmlRpcController.h
runListener, 408
XmlRpcController::XmlRpcControllerException, 328

XmlRpcControllerException, 328
XmlRpcControllerException
XmlRpcController::XmlRpcControllerException, 328