Home

Opal Documentation

Opal User Guides

The Opal Web Application User Guide is for users accessing to the Opal through a web browser.

The Opal R and DataSHIELD User Guide is for users accessing to Opal through a R programming language.

The Opal Python User Guide is for users accessing to Opal through a Python programming language.

The Opal Reporting User Guide is for users accessing to Opal through R markdown reports.

Opal Administrator Guides

The Opal Installation Guide is for system administrators who will install or upgrade Opal.

The R Server Installation Guide is for system administrators who will install a R Server to be used by Opal for running statistical analysis using R.

The Opal Configuration Guide is for system administrators who can use certain procedures in this guide to fine-tune the Opal configuration after the initial installation.

Concepts and Tutorials

What is Opal ?

Opal is OBiBa's core database application for biobanks. Participant data, once collected either from OBiBa's Onyx application, must be integrated and stored in a central data repository under a uniform model. Opal is such a central repository. Current Opal version can import, process, and copy data. Opal is typically used in a research center to analyze the data acquired at assessment centres. Its ultimate purpose is to achieve seamless data-sharing among biobanks.

For more information on Opal future, see Opal description on OBiBa.

Variables and Data

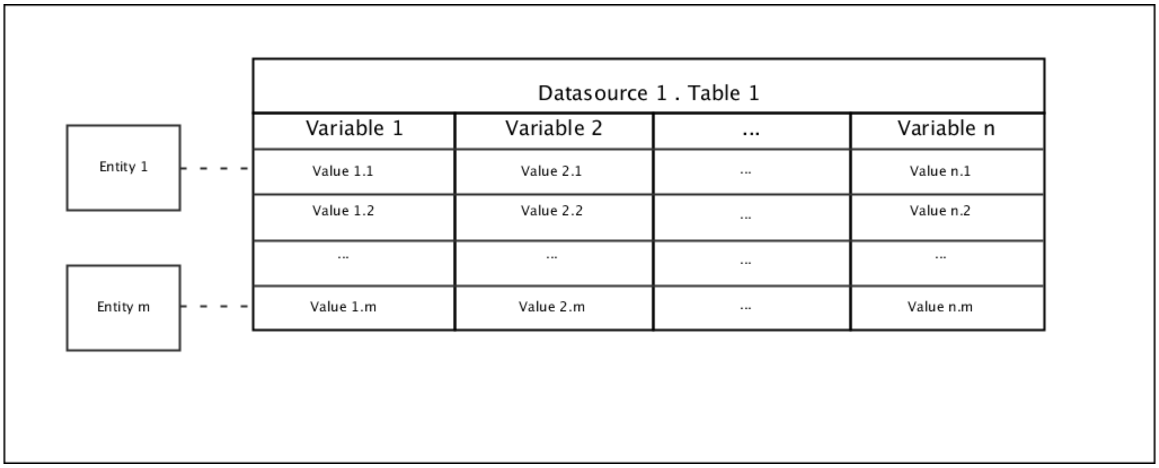
Overview

Introduction

The variables are organized in an abstract way, independently of the way they are persisted.

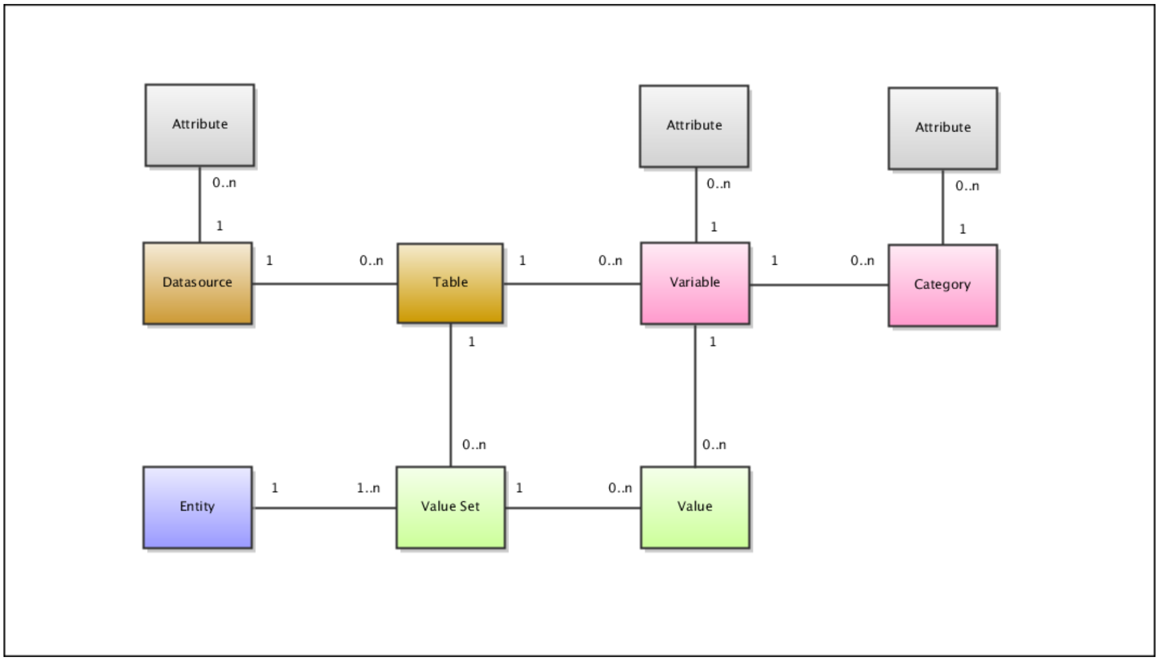
The following diagram presents a 'traditional' view of what is a table:

- the 'columns' are the variables,
- the 'rows' are the value sets for each entity,
- the 'cells' are the variable entity values.



Datasource 1 . Table 1			
Variable 1	Variable 2	...	Variable n
Value 1.1	Value 2.1	...	Value n.1
Value 1.2	Value 2.2	...	Value n.2
...	...	...	...
Value 1.m	Value 2.m	...	Value n.m

The following diagram shows the relationships between the different concepts:



Variables

Variables and Categories

A variable describes a set of values. The values of a variable are all of the same type. Possible value types are:

- `integer`
- `decimal`
- `text`
- `binary`
- `locale`
- `boolean`
- `datetime`
- `date`

A variable is about an entity, i.e. all the values for a variable are from the same entity type. Possible entity types are:

- `Participant`
- `Instrument`
- `...`

A category describe some of the possible values of a variable. A category is associated to one and only one variable.

Datasources and Tables

A variable is in one and only one table.

A table has several variables and is in one and only one datasource.

A datasource has several tables. A datasource is not a database: it can be persisted in a database, using different schema. It can also be persisted in a file in xml or Excel formats. It is important to understand that Opal separates the formal description of the variables from the way they are persisted. This gives to Opal a lot of versatility.

Attributes

Datasources, variables and categories have attributes. These attributes provide additional meta-information. An attribute is made of:

- a **namespace** (optional),
- a **name** (required),
- a **locale** (optional), that specifies in which language is the attribute value,
- a **value** (required even if null).

Example

Example of a variable `asked_age` which has the following attributes:

Name	Locale	Value
`label`	`en`	`What is your age ?`
`label`	`fr`	`Quel est votre age ?`
`questionnaire`		`IdentificationQuestionnaire`
`page`		`P1`

The variable `asked_age` has also some categories (with their attributes):

Name	Attributes

`888`	`label:en=Don't know`
	`label:fr=Ne sait pas`

`999`	`label:en=Prefer not to answer`
	`label:en=Préfère ne pas répondre`

Fully Qualified Names

Each of these elements has a short name. A fully qualified name will identify them uniquely:

- Datasource fully qualified name: `<datasource_name>`
- Table fully qualified name: `<datasource_name>.<table_name>`
- Variable fully qualified name: `<datasource_name>.<table_name>:<variable_name>`

The fully qualified name might be useful for disambiguation.

Examples

Following the example of the `asked_age` variable, its fully qualified name could be: `opal-data.IdentificationQuestionnaire:asked_age`

Derived Variables

A derived variable is a variable which values are computed using a script. This script is expressed using the Magma Javascript API.

Views

Opal deals with variables and values in tables. Views are here to:

- define a subset of a table, both in terms of variables and values,
- define a subset of many tables in terms of variables and values,
- define #Derived Variables that are to be resolved against 'real' ones.

These virtual tables are then manipulated just like standard tables (for instance they can be copied to a datasource).

Given one table:

Table1			
entity	**Var1**	**Var2**	**Var3**
1	Value 1.1	Value 2.1	Value 3.1
2	Value 1.2	Value 2.2	Value 3.2
3	Value 1.3	Value 2.3	Value 3.3

A view can be defined so that the resulting 'table' may be:

View1		
entity	**Var1**	**Var3**
1	Value 1.1	Value 3.1
3	Value 1.3	Value 3.3

or:

View2	
entity	**DerivedVar = function(Var1, Var2)**
1	function(Value1.1, Value2.1)
3	function(Value1.3, Value2.3)

Given Table1 above and the following table:

Table2			
entity	**VarA**	**VarB**	**VarC**
100	Value A.100	Value B.100	Value C.100
200	Value A.200	Value B.200	Value C.200
300	Value A.300	Value B.300	Value C.300

A view can also be a combination or a 'join' of both tables:

View3

entity	Var1	Var3	VarA	VarC
1	Value 1.1	Value 3.1		
3	Value 1.3	Value 3.3		
100			Value A.100	Value C.100
300			Value A.300	Value C.300

Data

Entities

The entities can be of different types:

- **Participant** (most common)
- **Instrument** (provided by Onyx)
- **Workstation** (provided by Onyx)
- **...** (any that might fit your needs)

Each entity has a unique identifier. An entity can have several value sets, but only one value set for a particular table.

Value Types

The following table gives more information about the textual representation of a value, given a value type:

Value Type	Value as a String
`integer`	The string value must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting integer has radix 10 and the supported range is [-2 ^{63}, 2 ^{63}-1].
`decimal`	As described by Java Double documentation.
`text`	As-is.
`binary`	Base64 encoded.
`locale`	String representation of a locale is `<language>[_<country>[_<variant>]]` (for instance `en`, `en_CA` etc.) where: • language: lowercase two-letter ISO-639 code. • country: uppercase two-letter ISO-3166 code. • variant: vendor specific code, see Java Locale.
`boolean`	True value if is equal, ignoring case, to the string "true".

`datetime`	Date times are represented in ISO_8601 format: "yyyy-MM-dd'T'HH:mm:ss.SSSZ" Supported input formats are (four digits year is required): - yyyy-MM-dd'T'HH:mm:ss.SSSZ - yyyy-MM-dd'T'HH:mm:ssZ - yyyy-MM-dd'T'HH:mmZ - yyyy-MM-dd'T'HH:mm:ss.SSSzzz - yyyy-MM-dd HH:mm:ss - yyyy/MM/dd HH:mm:ss - yyyy.MM.dd HH:mm:ss - yyyy MM dd HH:mm:ss - yyyy-MM-dd HH:mm - yyyy/MM/dd HH:mm - yyyy.MM.dd HH:mm - yyyy MM dd HH:mm
`date`	Dates are represented in ISO_8601 format: "yyyy-MM-dd" Supported input formats are (four digits year is required): - yyyy-MM-dd - yyyy/MM/dd - yyyy.MM.dd - yyyy MM dd - dd-MM-yyyy - dd/MM/yyyy - dd.MM.yyyy - dd MM yyyy

`point`	Point coordinates (longitude, latitude).
	Supported input formats are:
	- GeoJSON
	- JSON
	- Google Map
	`# GeoJSON`
	`{type: "Point", coordinates: [-71.34, 41.12]}`
	
	`# GeoJSON coordinates only: [lon,lat]`
	`[-71.34, 41.12]`
	
	`# JSON (different flavours of keys)`
	`{"lat" : 41.12,"lon" : -71.34}`
	`{"lat" : 41.12,"lng" : -71.34}`
	`{"latitude" : 41.12,"longitude" : -71.34}`
	`{"lt" : 41.12,"lg" : -71.34}`
	
	`# String, comma separated latitude and longitude (Google map like): lat,lon`
	`41.12,-71.34`

`linestring`	Array of point coordinates.
	Supported input format is GeoJSON:
	`# GeoJSON`
	`{type: "LineString", coordinates: [[22.2,44.1],[33.4,55.3],[32.12,44]]}`
	
	`# GeoJSON coordinates only`
	`[[22.2,44.1],[33.4,55.3],[32.12,44]]`

`polygon`	Array of shapes. A shape is a list of points. The last point must be equal to the first point.
	Supported input format is GeoJSON:
	`# GeoJSON`
	`{type: "Polygon", coordinates: [[ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0], [100.0, 1.0], [100.0, 0.0] ]]}`
	
	`# GeoJSON coordinates only: one shape polygon`
	`[`
	`  [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0], [100.0, 1.0], [100.0, 0.0] ]`
	`]`
	
	`# GeoJSON coordinates only: polygon with outter and inner shapes`
	`[`
	`  [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0], [100.0, 1.0], [100.0, 0.0] ],`
	`  [ [100.2, 0.2], [100.8, 0.2], [100.8, 0.8], [100.2, 0.8], [100.2, 0.2] ]`
	`]`

Value Sets and Values

A value is associated to a variable and is part of a value set. Each value set is for a particular entity and a particular table. An entity has a maximum of one value set in one table.

A value is always associated with a type and a data (or a sequence of data if the variable is `repeatable`).

Identifiers Mappings

Overview

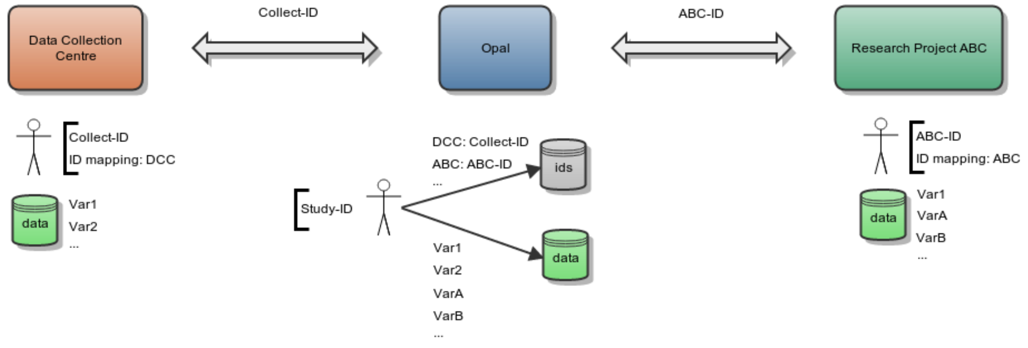
Introduction

Following the OBiBa paradigm of separation of concerns, the concept of "Identifiers Mappings" defines how to protect participant's privacy while exchanging data with Opal. The exchanges can be in both directions: imports and exports.

Participants privacy is ensured by not communicating private participant identifier (use of a shared key instead).

Participant Identifier Separation

The following diagram shows the different identifiers that can be assigned to one participant.



Opal *separates* the participant identifiers from the participant's data in two databases:

- the Opal identifier database will store the participant identifiers,
- the Opal data database will store anonymous participant's data.

One participant is identified in these two databases by a unique identifier which is the system identifier (usually the study identifier). Opal is able to find a participant from a given shared identifier.

Data Importation

The importation process is the following for one participant:

- if an identifiers mapping is provided, different importation strategies can apply:
  - each imported identifier must be mapped to a system identifier, otherwise the importation will fail,
  - each imported identifier must be mapped to a system identifier, otherwise the importation will ignore these data,
  - or when an imported identifier cannot be mapped to a system identifier, create a unique system identifier and map it to the imported one,
- else, no identifiers mappings is specified and therefore imported identifiers are considered to be system identifiers.

Data Exportation

The same identifiers mapping process can apply when exporting data. To avoid collusion between research projects requesting for data, each of them will be delivered data with participant's identifiers specific to them. This way "Research Project ABC" will not be able to put in common its Study data with "Research Project 123". It will only be possible if the Study allows it.

Integration with Onyx

Onyx is the OBiBa's solution for collecting participants data. Data exported by Onyx can be directly imported in Opal. If participant is assigned a different identifier in each data collection sites, then it will be required to define in Opal a identifiers mapping for each of these sites.

Data Harmonization with Opal

Overview

- Opal Application
  - What is Opal
  - Data Harmonization with Opal
  - Results and Benefits
- Data Harmonization Infrastructure
  - Opal in Biobank Networks
  - Data Harmonization across Biobanks
  - Biobanks Consortium Web Portal
  - Distributed analyses with DataSHIELD

Opal Application

What is Opal

Opal is a data integration software for Biobanks (or epidemiological studies w/o biosamples). Opal can be used to manage heterogeneous and anonymized data acquired from different sources and at different points of time. Opal can import, query and export data collected by Biobanks. Opal integrates with other softwares for analyzing (with R) and reporting (with BIRT).

Data Harmonization with Opal

Opal includes a comprehensive software infrastructure facilitating data harmonization as well as seamless and secure data-sharing amongst Biobanks.

To achieve effective data harmonization and querying of harmonized datasets between Biobanks, the steps are:

1. Set up Opal servers for each Biobanks and import relevant data sets,
2. Configure a harmonized description of data in each Opal server,
3. Set up a Mica server that is able to authenticate itself against each Opal server,
4. Run distributed queries on harmonized data sets.

Results and Benefits

When several Biobanks set up a network of Opals with the aim of harmonizing data, the benefits are:

- Individual-level data are hosted by the Biobank they belongs to,
- Each Biobank controls access rights to data in Opal.
- Consistent data access across Opal servers.

Collaborative research projects are highly facilitated when harmonizing data using Opal. Opal provides:

- Formal descriptions of harmonized data,
- Real-time availability of Harmonized dataset summaries from each Biobank,
- Real-time distributed statistical analysis through DataSHIELD method.

Opal is strongly integrated with Mica, a generic web portal for Biobank consortia. Through the Mica web interface, authenticated researchers can perform distributed queries on the content of each individual Biobank data collection hosted by Opal. Moreover, Opal implements the DataSHIELD method which enables individual-level data analysis across multiple Opal instances.

Data Harmonization Infrastructure

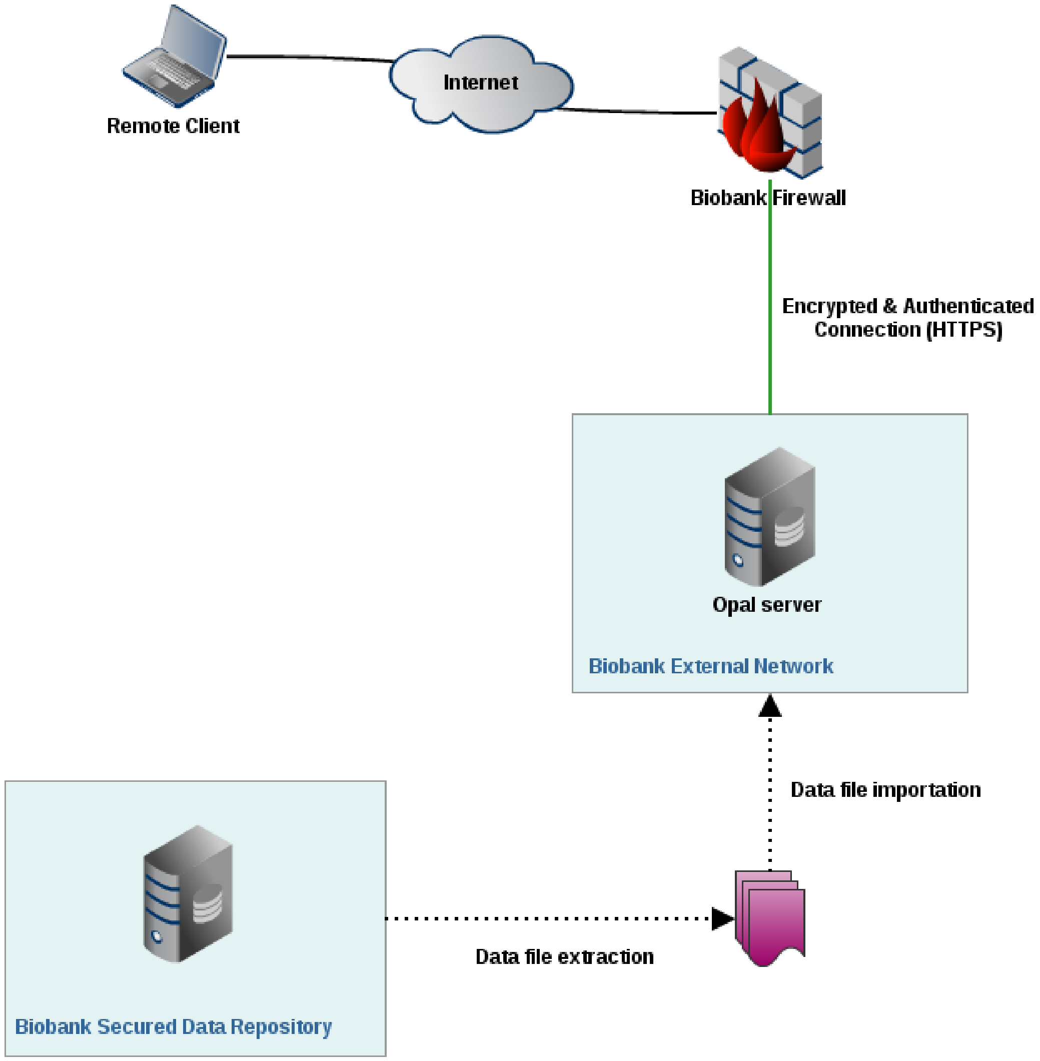
Opal in Biobank Networks

Opal is an application that runs on a server. Opal can be accessed through a secured connection (encrypted and authenticated). One possible network architecture to integrate Opal in Biobank infrastructure is the following:

- The *Biobank Secured Database Repository* is where the Biobank data is stored. Ideally it is not connected to any network and therefore data are imported in Opal using files.
- The *Biobank External Network* is a private network that hosts the Opal application and a database (running on the same server or on two different servers).

Opal ensures data access security through a variety of mechanisms:

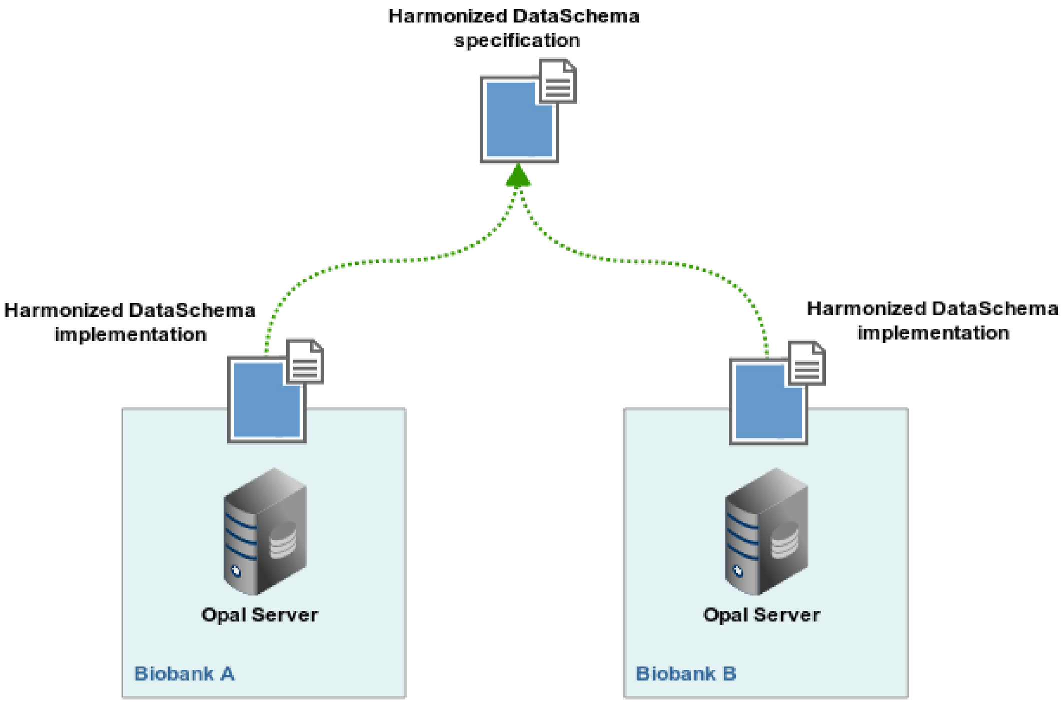
- The Opal server is hosted in a network that is protected by a firewall which only allows connections using encrypted (HTTPS) protocol through a specific port. Connections can also be restricted to specific remote clients.
- Opal application itself requires user authentication. Data hosted by Opal are subject to authorization (authenticated users can only see authorized data).
- Data are extracted from Biobank database as CSV files. These files are then imported in Opal database through the Opal application. There is no direct link between Opal and the Biobank Data Repository.



Data Harmonization across Biobanks

The aim of the data harmonization process is to transform biobank-specific data into a common format defined in the DataSchema and to access data in each Biobanks:

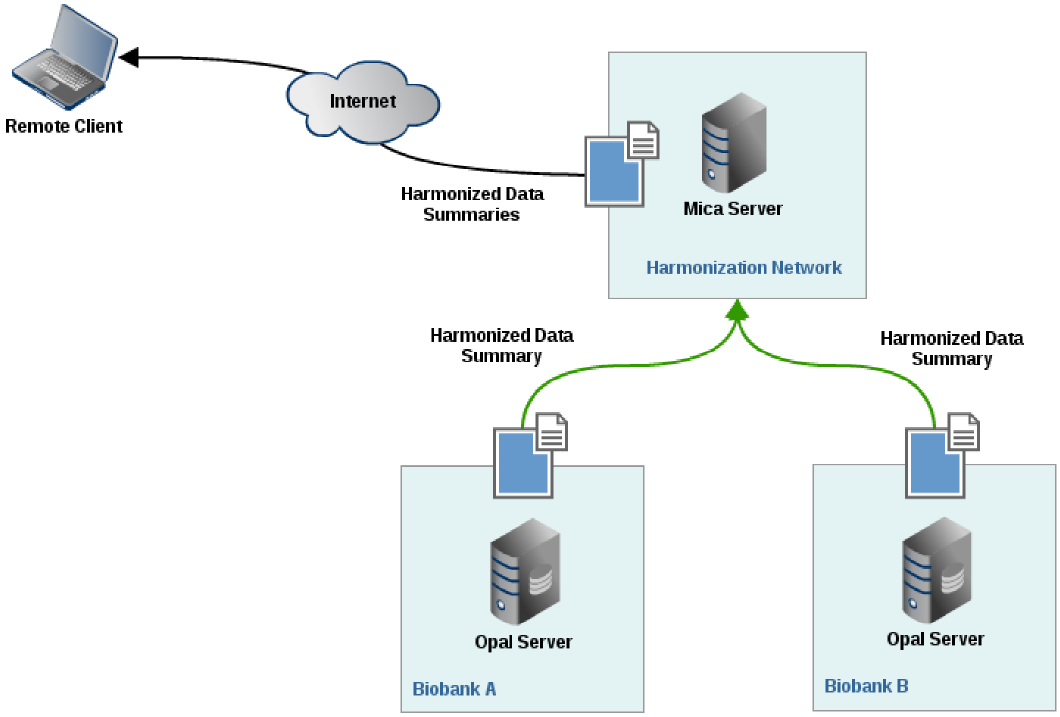
- The Biobanks have to agree on a Harmonized DataSchema, i.e. the description of the common data format,
- Each Biobank imports relevant datasets onto their dedicated Opal server,
- The Harmonized Data schema is uploaded in each Opal servers,
- Processing algorithms are then developed to derive biobank-specific variables into DataSchema format variables.


Harmonized DataSchema specification
Harmonized DataSchema implementation
Harmonized DataSchema implementation
Opal Server
Opal Server
Biobank A
Biobank B

Biobanks Consortium Web Portal

Mica is a web portal which aims at disseminating summary data from consortium members once it has been harmonized. A Mica server will connect and authenticate itself against each of the Biobanks Opal servers holding the harmonized datasets. In return Mica server will display in its web interface data summaries of harmonized variables (count, min/max, mean, stdv etc.) to the remote user.

> Note that when accessing harmonized data summaries:
>
> - The remote client never connects directly to any Opal server (Mica act as a broker),
> - Individual-level data are never extracted from Opal servers (data aggregations are computed in Opal).



Distributed analyses with DataSHIELD

DataSHIELD stands for Data Aggregation Through Anonymous Summary-statistics from Harmonized Individual-levEL Databases.

Some research projects demand very large sample size for detecting interactions. Such projects usually require pooling individual-level data from several studies to obtain this sample size. Important ethico-legal constraints often prevent or impede the pooling of individual level data.

DataSHIELD is a method by which an analysis of individual-level data from several sources can be done without actually pooling the data from these sources together. The process is described in a paper published in IJE. Through Mica web interface, distributed DataSHIELD queries can be run on any harmonized data sets hosted on Opal.

How to install and use Opal and DataSHIELD for Data Harmonization and Federated Analysis

1 Setting up an Opal and an R server

This section explains in details how to install and configure both Opal, R and a database application.

Opal will be used as the core data warehouse in your installation. Opal provides all the necessary tools to import, transform and describe data. Note that Opal is not a database: Opal needs to connect to a database application to store its data.

R provides a software environment for statistical computing and graphics. R will be used as a server: Opal will connect to this server to control remote accesses, push data to be analysed and retrieve results.

The instructions given assume that Opal, R and the database servers will be deployed on Ubuntu 14.04 LTS. These applications can also be installed on other Debian-like Linux distributions (Debian 7 for instance). Packages for Fedora-like Linux distributions (Fedora, CentOS 6 or 7) are also available: alternative installation instructions for these systems are provided as reference to the relevant installation guides.

Most of the following instructions and related information are taken from Opal Server Administrator Guide. Please refer to this online documentation for updated information.

1.1 Hardware Requirements

Although it is possible to install Opal, R and the database applications on different machines, this documentation assumes that these different software will be installed on the same host. As a consequence the hardware requirements must satisfy the combination of these three applications.

Component	Requirement
CPU	Recent server-grade or high-end consumer-grade processor
Memory (RAM)	8GB required, >= 16GB recommended
Disk space	5GB of free disk space required*

* The required disk space varies in function of the number of participants (and variables) in the datasets. Please use the following "rule of thumb" to evaluate your needs: 1 GB for the software + 4 GB/10000 participants.

Note that R is a single-threaded application and having CPUs with multi-cores will not make R computations faster: the performance of a single CPU core should be as good as possible.

Note also that R works in memory, then make sure the server has enough RAM for satisfying R needs without affecting too much the other applications. The recommended RAM size then should be adjusted depending on the size of the datasets and the type of statistical analysis that will be conducted (and the number of researchers running analysis simultaneously).

1.2 Installing Opal Server

This section is about installing both Opal and an associated database application.

1.1.1 Software Requirements

The following softwares must be installed on your server before you install Opal.

Note: At least one database management system must be installed. It can be either MongoDB, MySQL or both. Unless you have specific needs or constraints, MongoDB is recommended.

Software	Suggested Version	Use	Installation/Configuration
Java Runtime Environment	JRE 8.x	Java runtime environment - needed to run Opal	JRE 8 Ubuntu Installation Guide
MySQL	>= 5.5.x	Database management system - stores data imported into Opal	MySQL Database Configuration
MongoDB	>= 2.6.x	Database management system - stores data imported into Opal	MongoDB Database Configuration

1.1.1.1 Installation of Java

Using apt , you can install Java 8 via the following sequence of commands:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

Alternatively download Oracle Java RPM package.

1.1.1.2 Installation of the Database

One database server is required for Opal server to be able to operate. It is your choice to install MySQL or MongoDB. Due to the data schema used by Opal for storing data in the database, MongoDB is known to be better for large datasets.

1.1.1.2.1 MongoDB installation [recommended]

MongoDB is the recommended database engine.

See detailed MongoDB Ubuntu Installation Guide that can be summarized as follow:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
echo 'deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.0
multiverse' \
  | sudo tee /etc/apt/sources.list.d/mongodb.list
sudo apt-get update
sudo apt-get install -y mongodb-org
```

Alternatively see MongoDB RedHat/CentOS Installation Guide.

1.1.1.2.2 Installation of MySQL

MySQL database server can be installed using the following apt command:

```
sudo apt-get install mysql-server
```

Alternatively see MySQL RPM Repository instructions.

Finalise the installation by following the recommended MySQL server configuration.

1.1.2 Installation of Opal

Latest Opal server installation recommendations are available online in the Opal Installation Guide.

Once installed the Opal server is up and running (though not fully configured).

Prerequisite: The package apt-transport-https is required for OBiBa’s repository communicate through HTTPS. If you don’t have it installed on your system, install it via:

```
sudo apt-get install apt-transport-https
```

Then run the following commands to register the OBiBa Debian packages repository where Opal and other tools lies and install Opal server:

```
wget -q -O - https://pkg.obiba.org/obiba.org.key | sudo apt-key add -
echo 'deb https://pkg.obiba.org stable/' \
 | sudo tee /etc/apt/sources.list.d/obiba.list
sudo apt-get update
sudo apt-get install opal
```

Alternatively see OBiBa RPM repository instructions.

1.1.3 Configuration, Administration and Execution of Opal

Options for the Java Virtual Machine

Default configuration of Opal is usually not suitable for a production server. More specifically, the Opal application is allocated a maximum of 2G of RAM by default which could be not enough for operating on large datasets.

To increase the allocated memory edit the file /etc/default/opal and modify the value of JAVA_ARGS accordingly: - Xmx argument is to be changed for a higher value (at least 4G).

Then restart Opal server for making the new settings effective:

```
sudo service opal restart
```

Opal service log files

For troubleshooting, the log files to be inspected are located in /var/log/opal directory.

Opal offers many configuration possibilities (see Opal documentation for more details):

- Some are file-based (located in the folder /etc/opal)
- Others are accessible from the web application interface (located at https://<host>:8443)

1.2 Installing R Server

The R server consists of several pieces of software:

- R, is the R language interpreter,
- Rserve, is an R package that allows to start an R session from a distant connection,
- R Server Admin, is a Java-based application that allows to start and stop Rserve from a distant connection.

Latest R server installation recommendations are available online in the R Server Installation Guide.

1.2.1 Software Requirements

Software	Suggested Version	Use	Installation/Configuration
Java Runtime Environment	JRE 8.x	Java runtime environment - needed to run R Server Admin	JRE 8 Ubuntu Installation Guide
R	>= 3.1.x	Statistical analysis engine	http://cran.r-project.org

1.2.2 Installation of R Server

A Debian package is provided to conveniently install all the softwares required to build an R server that is ready to be used by Opal.

The default R provided by the Linux distribution is not the latest one: the R Debian repository is to be added first.

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E084DAB9
echo 'deb https://cran.rstudio.com/bin/linux/ubuntu trusty/' \
  | sudo tee /etc/apt/sources.list.d/cran.list
sudo apt-get update
```

Then the R server installation is performed using the following commands:

```
sudo apt-get install opal-rserver
sudo service rserver restart
```

Once done, the R server should be up and running. Default settings are usually enough for Opal to connect to this server.

Alternatively see CRAN RPM repository and OBiBa RPM repository instructions.

1.3 Security Configuration

Several servers are running on your system, but only a limited number of services should be exposed. The communication to these services must be encrypted and can be restricted to a limited set of clients.

The following firewall rules must be applied:

1. Allow HTTPS connections to Opal through the port 8443 (this port number can be configured; a reverse proxy can also be setup),

2. Allow SFTP connections to Opal through the port 8022 (this firewall rule is optional, as file upload can also be performed through HTTPS),

3. Allow a limited set of external IP addresses to connect to the host (typically for the DataSHIELD analysis server (see below) and administrators).

The other services (R and database servers) must not be exposed. Opal server will access them locally.

Advanced security setting that can be considered is to restrict the supported encryption protocols and algorithms used by Opal server when communicating through HTTPS . This can be achieved at Opal and/or Java levels. See Opal server documentation for more details.

2 Infrastructure Testing

This section are instructions for cohorts to set up a test dataset on their server that we can test against.

The tests you will perform on Opal are quite simple:

1. You will upload a test dataset in Opal. This will ensure that Opal is correctly configured, that it is properly connected to its database, and that it can be administered through its web interface.
2. You will make some calculations on this dataset from a distant computer. These calculations will be made using R, and success of this testing will guarantee that R Server Admin is properly installed.

2.1 Getting the Test Dataset

The test dataset (generated data) to be downloaded is LifeLines.sav.

2.2 Setup the Data for Testing

Opal's administrative web interface can be accessed with any modern web browser by pointing it to:

```
https://<host>:8443
```

where <host> is the IP address or the host name where Opal is located on the network.

2.2.1 Configuring the Databases

Before proceeding, Opal storage databases have to be configured. If you already done this, go to step 2.2.2 .

Otherwise, for being fully operational, Opal requires to register two databases (that can be located in the same database server): one for the participant identifiers and one for their data.

First, prepare the databases in the database server:

- one with name opal_ids,
- one with name opal_data.

Create a user with administrator privileges on these databases. See MongoDB Database Creation or MySQL Database Creation instructions for more details.

Then databases registration is done as follow:

1. Login as an administrator in Opal web interface,
2. A "Post-Install Configuration" page is displayed allowing to set up the Opal databases,
3. In the "Identifiers Database" section, click on "Register" and select the database type of your choice ("SQL" or "MongoDB").
4. Input information for the database: provide the name of the database along with connection information (url, username and password) and save the settings.
5. Do the same for the "Data Databases" (the database url must be different from the one for the identifiers).

2.2.2 Uploading data for testing

Some test data will be imported from a file. This file needs to be accessible from the Opal server. For that purpose, Opal has a "file system" where the data files can be uploaded.

From the "Dashboard" page, click on "Manage Files":

1. Navigate to the directory where the data file will be uploaded.
2. Click on the "Upload" button; this will open a "File Upload" window which allows you to select a file to upload from your computer.
3. Click on "Choose File" and select the LifeLines.sav file you have saved at section 2.1 . Once done, click on the "Upload" button.

Note

Since LifeLines.sav is an SPSS file, it includes both the data dictionary (i.e. variable coding and labels) and the participant data.

2.2.3 Create a Project and Import Data

In Opal, a project is the workspace for managing data. It is required to create a project before importing data into Opal.

1. Go to the "Projects" page.
2. Add a Project by clicking on the "Add Project" button. Then, in the "Add Project" popup window,
    1. Enter test in the name field,
    2. Select the database you created in step 2.2.1 as the project’s data store,
    3. Save the project.

Then import data into this project.

1. Go to the test project page, tables section,
2. Click on the "Import" button to open the "Import Data" window.
3. For the "Data Format" drop-down, select "SPSS" option and click on the "Next" button.
4. Under "Data File", click "Browse" to select (tick) the LifeLines.sav file, then click on "Select".
5. Click on the "Next" button so that you skip the "Configure data import" step.
6. Tick the checkbox to the left of the LifeLines table and click on the "Next" button.
7. You can review the data for the table LifeLines to be imported, then click on the "Next" button.
8. Keep the default setting for data file archiving and click on the "Finish" button.

You can follow the import task progress by going to the tasks section of the project page. Once importation is completed successfully, the LifeLines table should appear in the tables section of the project page.

2.3 Test the R Server with Test Data

You will use the data you imported in the previous section in order to test whether the R Server is correctly installed and works correctly with Opal.

From the shell prompt of the server, start the R console:

```
R
```

The following R script should be executed without errors (make sure to change in this script the administrator’s password with the one of your server):

```
# Load Opal R library
require('opal')

# Then, create an opal object with the login information
# Change the login credentials and url with the appropriate values!
o <- opal.login(username='administrator', password='password',
url='https://localhost:8443')

# To verify if the connexion with Opal works,
# get the list of all projects
opal.datasources(o)

# Assign the content of the LifeLines table (from the test project)
# into a data frame in a R session of the R server
opal.assign(o,'D','test.LifeLines', missings = TRUE)

# Get the summary of this data frame from the remote R session
opal.execute(o,'summary(D)')

# Terminate the remote R session
opal.logout(o)
```

The expected output result from the data frame summary command is:

```
GESLACHT    GEWICHT           LENGTE      HEALTH17A1 HEALTH17B1 HEALTH17D1
 1:2551   Min.   : 31.00   Min.   :144.2   1: 425     1:   0     1:4666
 2:2473   1st Qu.: 66.94   1st Qu.:172.2   2:4599     2:5024     2:   0
          Median : 76.08   Median :178.5              3:   0     3: 150
          Mean   : 76.20   Mean   :178.6              4:   0     4: 208
          3rd Qu.: 85.42   3rd Qu.:184.9                         5:   0
          Max.   :130.68   Max.   :210.5
      DBPa        SMK11    SMK31        SMK4A1       SMK4A21
 Min.   : 39.00   1: 997   1: 811   Min.   : 0.000   1: 708
 1st Qu.: 71.00   2:4027   2:4213   1st Qu.: 0.000   2:  58
 Median : 80.00                     Median : 2.000   3:4258
 Mean   : 79.78                     Mean   : 3.915
 3rd Qu.: 88.00                     3rd Qu.: 7.000
 Max.   :126.00                     Max.   :24.000
```

3 Install DataSHIELD for Federated Analysis

DataSHIELD is the platform that allows federated analysis to be carried out on your data. We assume in this section there is an existing Federated Database Network (FDN) with a central DataSHIELD analysis server. This section explains how to set up DataSHIELD on your data server and to connect this server to the existing FDN.

3.1 Install DataSHIELD packages

Each server in the network must be configured the same way so that same computation is done in each Opal for one client request. This is done by using the DataSHIELD-R packages repository.

To install these packages, follow these steps:

1. Go to the "Administration" page, and click on "DataSHIELD" section.

2. Click on "Add Package".

3. Leave the default option: “Install all DataSHIELD packages” and click "Install"

4. The DataSHIELD packages should appear in the list and the Methods section should also be populated with entries.

3.2 Create a user

Now we need to create a user account for the analysis server to be able to run a test analysis against your server:

1. Go to the "Administration" page, and click on "Users and Groups" section.

2. Click on "Add User".

3. Select "Add user with certificate".

4. Give the user the name “FDN_user” (usually this username is provided by the project administrator and reflects the name of the FDN your server has to connect to)

5. Ask the project administrator for the analysis server’s certificate

6. Paste the following certificate into the box and click "Save"

3.3 Set DataSHIELD Permissions

The FDN user must be given permission to access the server using DataSHIELD.

1. Go to the “Administration” page, and click on “DataSHIELD” section.

2. Scroll down the page and Click on “Add Permission”.

3. Select “Add user permission”.

4. Type “FDN_user” in the name field.

5. Leave the default selection of “Use” and click on “Save”.

3.4 Set Project Permissions

Now it is necessary to set up the permissions for the user on the LifeLines table in the test project.

1. Go to the “Projects” page and click on the test project.
2. Go to the tables section and click on the LifeLines table.
3. Click on the “Permissions” tab.
4. Click on “Add Permission”.
5. Select “Add user permission”.
6. Type “FDN_user” in the name field.
7. Leave the default value of “View dictionary and summaries” and click on “Save”.

3.5 Test the Functionality

Contact the project technical support who will check that they can run a summary analysis on the test dataset.

4. Transfer Study-specific Data into Opal

4.1 Prepare Study-specific Datasets to be Imported into Opal

4.1.1 Ensure the Datasets include needed information

4.1.1.1 Select Study-specific Variables

For each study, a list of study-specific variables will be selected based on the mapping protocol developed by the harmonization working group. The mapping protocol includes all study-specific variables required to generate the Dataschema variables (common core variables to be generated for the harmonized dataset).

4.1.1.2 Create new Dataset

The local team should develop a new dataset with the required variables and data. The naming of the dataset should indicate the study and data collection event it belongs to (e.g. baseline, wave 1, etc…). Each variable should include a name, clear label, and category codes and labels.

4.1.2 Manage Data Inconsistencies and Create Clean Datasets

4.1.2.1 Run Descriptive Statistics

Data distribution and variables' associations should be tested to ensure data quality. Data should be checked for odd distributions (e.g. 92% missing values for income), impossible ranges (e.g. sleeping more than 24 hours per day), and contradictory values (e.g. age of onset of diabetes is higher than the actual age of participant). The data should also conform to the questionnaire flow taking into consideration all skip patterns.

4.1.2.2 Manage Problematic Values and Document Decisions

Management of the problematic values is context specific. It is recommended that a local group reviews each case and recommends solution strategies. All data cleaning decisions should be transparent and well documented to be provided to the harmonization working group.

4.2 Import Study-specific Datasets into Opal

You will now import study-specific datasets into Opal. The procedure is akin to the one you did in 2.2.2 — 2.2.3 for the test dataset.

4.2.1 Datasource Type

While there are more than two different types of datasource, you will deal here only with the two you are likely to use for variable and data import namely:

- CSV is a "delimiter separated values" text file format. First row are the variable names, subsequent rows are the participant values. First column is expected to be the participant identifier. A CSV file can be imported as-is but the variables will be considered as being of text type, unless the data dictionary is prepared before the data import (as explained in the 4.3 section).

- SPSS source file must be a valid non-compressed binary file with a .sav extension. In Opal an SPSS file represents a table and its variables are used as the table's data dictionary. An Opal compatible SPSS file must have its first variable represent the identifiers. If this is not the case, before a file import, the identifier variable must be moved to the first position of the SPSS variable sheet.

For more information about the available file based formats, see Opal Datasource Types documentation.

4.2.2 Importing the Data

This is exactly as done in 2.2.2 - 2.2.3 except for the obvious variations: the name of the project, the format of the data file you are importing (either SPSS .sav files or CSV .csv files), and the path to the file.

4.3 Prepare Data Dictionary

Detailed information for each variable should be documented in a structured way to enhance comprehension and highlight existing heterogeneity across different studies. Required information for each study variable is presented in table 1. Data dictionary information can be updated directly in Opal or in Excel.

Table 1: Information to be documented for each study-specific variable

Field	Definition
Table	Name of the dataset the variable is associated with
Variable name	Name of the variable
Label	Short description of the variable specifying its content (e.g. Type of diabetes) Further information can be added in the description field.

Description	Additional information about the variable such as: • For variables collected by questionnaire, the question itself or any relevant information about the variable (e.g. Have you ever been told by a doctor that you had diabetes?) • For variables about physical/laboratory measures, any relevant information describing the context of measurement (e.g. self-reported measure, measure by a trained professional) or related to the protocol (e.g. measure taken when the participant is at rest) • For derived or constructed variables, any relevant information about the derivation or construction of the variable (e.g. MMSE total score, total energy in Kcal per day derived from diet questionnaire)
Value type	Type of variable: • Decimal (numerical values with a fractional component) • Integer (numerical values without a fractional component) • Text (alphanumerical values) • Date (values written in a defined date format) • Datetime (values written in a defined date and time format) • Boolean (two possible values (usually denoted true or false)) (e.g. Type of diabetes has an integer value type: 1, 2, 3, 8, 9)
Unit	Measurement unit of the variable (e.g. cm, mmol/L)
Additional information for categories	
Category code	Value assigned to each variable category (e.g. Type of diabetes has 5 categories: 1, 2, 3, 8, 9)
Category label	Short description of the category (e.g.: 1: Type 1 diabetes 2: Type 2 diabetes 3: Gestational diabetes 8: Prefers not to answer 9: Missing
Missing	Code assigned to each category identifying it as a missing value (e.g.: 1: Type 1 diabetes (Not missing = 0) 2: Type 2 diabetes (Not missing = 0) 3: Gestational diabetes (Not missing = 0) 8: Prefers not to answer (Missing = 1) 9: Missing (Missing = 1))

4.3.1 Updating Data Dictionary in Opal

Adding data dictionary information such as label, description and categories or changing value type of a variable can be done directly in Opal by following the steps below:

1. Go to the project page, tables section,
2. Go to the table page, then add a new variable or go to the variable page to be modified,
3. Edit variable properties (note that the value type cannot be modified if the table has data),
4. Edit the categories (add, update or remove categories),
5. Edit the attributes (label, description…).

4.3.2 Updating Data Dictionary from Excel

Data dictionary information can also be updated from an Excel file. This can be achieved by first downloading the excel data dictionary, then changing the information and finally uploading it in Opal. This method is usually more efficient when dealing with a large number of variables. Below are the steps:

1. Go to the project page, tables section,
2. Go to the table page,
3. Click on “Download” and select “Download dictionary”,
4. Open the downloaded Excel file:
   - In the “Variables” sheet, there is one row per variable and one column per property/attribute,
   - In the “Categories” sheet, there is one row per variable’s category.
5. Modify the information of the Excel file: do not modify the column names, make sure there are no duplication of variable/category rows and note that the value type cannot be modified if the table has data,
6. Update the table data dictionary with this Excel file:
   - Go to the project page, tables section,
   - Click on “Add Table” and select “Add/update tables from dictionary”,
   - Select the the Excel file (you need to upload it into Opal first), and click “Next”,
   - Review the changes, select the table to be added or updated, and click “Finish”.

5. Harmonisation and Federated Analysis

This section describes the steps that are necessary to implement harmonisation algorithms on the raw data to produce an harmonised dataset. This is the dataset that will be used for the federated analysis.

5.1 Harmonisation

Harmonisation requires the consortium agree on algorithms that, when applied to each dataset, will result in a common set of variables across all studies in the consortium. These algorithms are then coded in JavaScript on the server. Since writing the code in JavaScript is specialist knowledge, this can initially be done by the Project’s harmonisation team. To do this they will need a username that has sufficient permissions to set up the algorithms without being able to see the individual level data (higher privileges can be granted by the study).

5.1.1 Set up Harmonisation User

The credentials for the Harmonisation user are added as follow:

1. Go to the “Administration” page, and click on “Users and Groups” section,
2. Click “Add User”, and select “Add user with password”,
3. Give the user the name: Harmonisation ,
4. Choose a password,
5. Inform the project technical support of the password (password can be changed by the user afterwards).

5.1.2 Give Harmonisation User Permissions

The Harmonisation user must be able to see the data dictionary and summaries of the study-specific table:

1. Go to the project page, tables section,

2. Go to the study-specific table page,
3. On the “Permissions” tab, click “Add Permission” and select “Add user permission”,
4. In the name field, type Harmonisation ,
5. Leave the default permission of “View dictionary and summaries” (depending on the agreement with the study, the permission “View dictionary and values” should be chosen instead), and save.

Then the Harmonisation user should be granted the permission to create a view in this project:

1. Go to the project page, tables section,
2. On the “Permissions” tab, click “Add Permission” and select “Add user permission”,
3. In the name field, type Harmonisation ,
4. Leave the default permission of “Add table”, and save.

The view that will be added by the Harmonisation user will transform the variables of the study-specific table into the harmonised variables (without compromising the individual level data).

5.2 Federated Analysis

Once the harmonised dataset is ready, users will need to be given permission to run analyses on your data.

5.2.1 Add a Datashield User

This section only needs to be completed for new users:

1. Go to the “Administration” page, and click on “Users and Groups” section,
2. Click “Add User”, and select “Add user with certificate”,
3. Give the user the name supplied by the project technical support,
4. Paste the certificate for that user (supplied by the project technical support) and save.

New users will need to be given permission to connect via DataSHIELD:

1. Go to the “Administration” page, and click on “DataSHIELD” section,
2. Click “Add Permission”, and select “Add user permission”,
3. Type the name chosen above in the name box,
4. Leave the default selection of “Use” and save.

5.2.2 Give users permission to analyse data

Now it is necessary to set up the permissions for each user on the project:

1. Go to the project page, tables section,
2. Go the the harmonised table page,
3. On the “Permissions” tab, click “Add Permission” and select “Add user permission”,
4. Enter the name of the user,
5. Leave the default permission of “View dictionary and summaries”, and save.

Opal Server Administrator Guide

Contents of this Guide

Introduction

Opal is an application that runs on a server. Users access Opal via the web or on client workstations via a secure shell connection over a network.

This guide is for whoever will set up Opal--typically, a system administrator.

The guide covers hardware and software requirements and includes procedures for deploying Opal on a server and setting up the client workstations.

When requirements are met, administrators can follow:

1. the Opal Installation Guide,
2. the R Server Installation Guide,
3. and the Opal Configuration Guide.

Opal package is available in different format:

Type	Package Repository
Debian	Debian Repository
RPM	RPM Repository
Zip	Zip Repository
Docker	Docker Repository

Requirements

Hardware Requirements

Server Hardware Requirements

Component	Requirement
CPU	Recent server-grade or high-end consumer-grade processor
Memory (RAM)	Minimum: 4 GB Recommended: >8 GB
Disk space	Minimum: 160 GB Rule of thumb calculation: 10 GB for operating system + 4 GB/10000 participants

Client Hardware Requirements

Component	Requirement

CPU	Low-end consumer-grade processor

Software Requirements

Server Software Requirements

You must install the following software on the Opal.

Software	Suggested Version	Download Link	Use	Installation/Configuration
Java	>= Java 8.x	http://www.oracle.com/technetwork/java/javase/downloads/index.html	Java runtime environment - needed to run Opal	Java 8 Installation Guide Java 8 Ubuntu Installation Guide
MySQL	>= 5.5.x	http://www.mysql.com/downloads/mysql/	Database management system - stores data imported into Opal	MySQL Database Administration
MongoDB	>= 2.6.x	http://www.mongodb.org/downloads	Database management system - stores data imported into Opal	MongoDB Database Administration
R	>= 3.0.x	http://cran.r-project.org	Statistical analysis engine	R Server Installation Guide

At least one database management system must be installed. It can be MongoDB, MySQL or both.

MySQL, MongDB and R can be installed on a server machine different from Opal server.

Client Software Requirements

- To access Opal via the web you must have a recent web browser such as Firefox 3.6 or Chrome 8.0, Internet Explorer 9.0.
- To access Opal via SFTP use any graphical clients such as FileZilla or the Firefox add-on FireFTP.

Opal Installation Guide

Contents of this Guide

Introduction

Opal is an application that runs on a server. Users access Opal via the web or on client workstations via a secure shell connection over a network.

This guide is for whoever will set up Opal--typically, a system administrator.

Installing Opal

Opal is distributed as a Debian package, a RPM package and as a zip file. Installing the Debian package is recommended on Debian-like Linux systems and the RPM package on RPM-based Linux distributions (e.g. CentOS, Fedora, Red Hat).

The resulting installation has default configuration that makes Opal ready to be used. Once installation is done, see how Configuring Opal.

Installation of Opal Debian package (recommended)

Opal is available as a Debian package from OBiBa Debian repository.

Download Opal Debian package

To proceed installation, do as follows:

1. **Install Debian package**. Follow the instructions in the repository main page for installing Opal.

2. **Manage Opal Service**: after package installation, Opal is running: see how to manage the Service.

Installation of Opal RPM package (recommended on RPM-based Linux distributions)

Opal is available as a RPM package from OBiBa RPM repository.

Download Opal RPM package

To proceed installation, do as follows:

1. **Install RPM package** following the instruction RPM repository main page for installing Opal.

2. **Change default administrator password** by following the steps under File Based User Directory. This security requirement is due to RPM's lack of interactivity during installation.
3. **Manage Opal Service**: after package installation, Opal is running: see how to manage the Service.

Installation of Opal Zip distribution

Opal is also available as a Zip file.

Download Opal Zip package

To install Opal zip distribution, proceed as follows:

1. **Download Opal distribution**
2. **Unzip the Opal distribution**. Note that the zip file contains a root directory named `opal-server-`*`x`*` . `*`y`*` . `*`z`* (where *x*, *y* and *z* are the major, minor and micro releases, respectively). You can copy it wherever you want. You can also rename it.

3. **Create an OPAL_HOME environment variable**
4. **Separate Opal home from Opal distribution directories (recommended)**. This will facilitate subsequent upgrades.

Set-up example for Linux

```
mkdir opal-home
cp -r opal-server-x/conf opal-home
export OPAL_HOME=`pwd`/opal-home
./opal-server-x/bin/opal
```

Set-up example for Windows

```
C:\>md opal-home\conf
C:\>copy opal-server-x\conf opal-home\conf
C:\>setx OPAL_HOME "c:\opal-home"
C:\>opal-server-x\bin\opal
```

5. **Launch Opal**. This step will create/update the database schema for Opal and will start Opal: see Regular Command.

Upgrading from Opal 1.x

Opal 2.x is a major upgrade. Upgrade will mainly affect the configuration files. Variables, data and identifiers stored in the databases will not be modified.

> As a general rule, always backup the configuration files and the databases before upgrading. Make sure also you can restore them!

Opal 1.x Backup

In Opal 1.x, the configurations files are all located in the directory **OPAL_HOME/conf**.

The main Opal data database also contains the keystores and the permissions.

Here is a short script to backup Opal data, configuration and filesystem.

> This script does not backup data that are stored in databases. For information on backups of databases see MySQL Backup and Recovery.

```
#!/bin/bash

DATE=`date +%Y%m%d-%H%M%S`
DIR=backups/$DATE

# Change this to the path that is relevant for your install
OPAL_HOME=/var/lib/opal
mkdir -p $DIR/conf
cp -r $OPAL_HOME/fs $DIR/fs
cp -r $OPAL_HOME/conf/* $DIR/conf
```

Upgrading from Opal <=1.14.x

Opal with version older than 1.15.x cannot be upgraded directly to 2.x. You should then consider the following options:

- upgrade to Opal 1.15.x before upgrading to Opal 2.x,
- or

- export your data in Opal archive files (XML files in a zip),
- download the views,
- make sure you can restore key pairs and certificates,
- list the permissions,
- install 2.x from scratch (including a new database),
- import the Opal archive files,
- reinstate the views, the permissions, the users, the key pairs and certificates

Upgrading from Opal 1.15.x

Upgrade from Opal Debian Package

Opal 2.x requires java8. Make sure java8 is the system default java before upgrading **opal** Debian package.

When installing the Opal Debian package, the package manager will ask for keeping the modified configuration files. Answer **N (keep your currently-installed version)**, Opal will take care of upgrading the content of these configuration files.

```
sudo apt-get install opal
```

Upgrade from Opal Zip distribution

Opal 2.x requires java8. Make sure java8 is installed and use it for executing opal:

- either set the system default java to be java8 (recommended),
- or modify the **opal** executable in **OPAL_DIST/bin** so that java8 is used.

Before starting Opal 2.x, copy some files from the Opal 2.x distribution to the **OPAL_HOME/conf** directory:

- copy the file **logback.xml** (this will ensure that the log messages will be written in **OPAL_HOME/log** files instead of the console),
- copy the file **newrelic.yml** (to not have a error message from New Relic).

Upgrade R Server

Opal 2.x is able to start/stop a remote R server. To achieve that you must install the R Server Admin application and therefore remove any R server that would have been installed.

Upgrading from Opal 2.4.8

To safely upgrade Opal <= 2.4.8 from a RPM package only, please remove the current installation as shown in the steps below before installing the new version. Releases following 2.4.8 address a packaging bug jeopardizing data folders upon removal:

```
sudo /etc/init.d/opal stop
sudo rpm -e opal-server --noscripts
```

Please note that the precautionary steps above do not apply to Debian install and packages.

Configuring Opal

Default configuration makes Opal ready to be used but you should consider to configure Opal to match your needs:

1. **Configure the Users Authentication Directories**: see User Directories Configuration,
2. **Configure third-party servers and other properties**: see Opal Main Configuration File,
3. **Configure databases**: see Databases Administration.

Executing Opal

Launching Opal Server

Service

When Opal is installed through the Debian package or the RPM package, Opal server can be managed as a service.

Service Environment

Options for the Java Virtual Machine can be modified if Opal service needs more memory. To do this, modify the value of the environment variable `JAVA_ARGS` in the file `/etc/default/opal`.

Service Script

Main actions on Opal service are: `start`, `stop`, `status`, `restart`. For more information about available actions on Opal service, type:

```
service opal help
```

Service Logs

The Opal service log files are located in `/var/log/opal` directory.

Manually

The Opal server can be launched from the command line. The environment variable `OPAL_HOME` needs to be setup before launching Opal manually.

Command Environment

Configure the following environment variables:

Environment variable	Required	Description

`OPAL_HOME`	yes	Path to the Opal "home" (configuration and file system) directory.
		For example (Windows):
		`C:\opal-home`
		Note: `OPAL_HOME` should point to the location (the parent) of the `conf` directory.

`JAVA_OPTS`	no	Options for the Java Virtual Machine.
		For example:
		`-Xmx4096m -XX:MaxPermSize=256m`
		To change the defaults update:
		`bin/opal` or `bin/opal.bat`
		Note: If there is not enough memory allocated to the JVM Opal may run slow and/or run out of memory running commands using large data sets.

Regular Command

Make sure Command Environment is setup and execute the command line (`bin` directory is in your execution `PATH`)):

- Windows console:

  ```
  opal.bat
  ```

- Unix shell:

  ```
  opal
  ```

Executing this command upgrades the Opal server and then launches it.

Command Logs

The Opal server log files are located in `OPAL_HOME/logs` directory. If the `logs` directory does not exist, it will be created by Opal.

Using Opal

Accessing Opal from the Web

To access Opal with a web browser the following urls may be used (port numbers may be different depending on HTTP Server Configuration):

- `http://<servername>:8080` will provide a connection without encryption,
- `https://<servername>:8443` will provide a connection secured with ssl.

See Opal Web Application User Guide for more details.

Troubleshooting

Always check the logs of Opal to know what went wrong: either Service Logs or Command Logs depending on how Opal was launched.

Opal Plugins Installation Guide

Contents of this Guide

Introduction

From Opal 2.9.0, new services can be added as plugins discovered at runtime. This guide specifies how to install and configure Opal plugins.

Repository

Opal plugins available are:

Name	Type	Description	Depends	API
`jennite-vcf-store`	`vcf-store`	Stores the genotypes in Variant Call Format (VCF) files (binary flavor, BCF, is also supported). VCF/BCF files can be downloaded filtered by participant phenotype criteria.	`bcftools`, `tabix`, `bgzip` executables , available from bcftools and htslib download page. Minimum version is 1.2.	VCF Store Plugin API
`opal-search-es`	`opal-search`	Opal search engine based on Elasticsearch 2.4. Can be used embedded in Opal (default) or configured to connect to an Elasticsearch cluster.	No dependencies.	Search Plugin API

Installation

All plugins are to be deployed as a directory at the following location: **OPAL_HOME/plugins**.

Manual Installation

Available plugins can be downloaded from OBiBa Plugins Repository. The manual installation procedure should be performed as follow:

1. **Download the plugin** of interest (zip file) from OBiBa Plugins Repository,
2. **Unzip plugin package** in **OPAL_HOME/plugins** folder. Note that the plugin folder name does not matter, Opal will discover the plugin through the *plugin.properties* file that is expected to be found in the plugin folder.
3. **Read the installation instructions** (if any) of the plugin to identify the system dependencies or any other information,
4. **Restart Opal**.

Administration Page Installation

The plugins can also be managed (installation, upgrade, removal) from a dedicated administration page. See Plugins Administration documentation.

Debian Package Installation

Installing a Opal plugin from a Debian package will also automatically install the system dependencies (if any) of the plugin, which can be very convenient. Debian package will also take care of the plugin upgrade.

Opal plugins are available as Debian packages from OBiBa Debian repository.

Download Opal Plugins Debian packages

To proceed installation, do as follows:

1. **Install Plugin Debian package**. Follow the instructions in the repository main page for installing Opal plugin.
2. **Configure Plugin** (if necessary), by editing the *site.properties* files,
3. **Restart Opal**.

RPM Package Installation

Installing a Opal plugin from a RPM package will also automatically install the system dependencies (if any) of the plugin, which can be very convenient. RPM package will also take care of the plugin upgrade.

Opal plugins are available as RPM packages from OBiBa RPM repository.

Download Opal Plugins RPM packages

To proceed installation, do as follows:

1. **Install Plugin RPM package**. Follow the instructions in the repository main page for installing Opal plugin.
2. **Configure Plugin** (if necessary), by editing the *site.properties* files,
3. **Restart Opal**.

Configuration

The **OPAL_HOME/plugins** folder contains all the Opal plugins that will be inspected at startup. A plugin is enabled if it has:

- A valid *plugin.properties* file,
- In case of several versions of the same plugin are installed, the latest one is selected.

The layout of the plugin folder is as follow:

```
OPAL_HOME/
 plugins
     <plugin-folder>
         lib
            <plugin-lib>.jar
         LICENSE.txt
         README.md
         plugin.properties
         site.properties
```

Inside the plugin's folder, a properties file, *plugin.properties*, has two sections:

- The required properties that describe the plugin (name, type, version etc.)
- Some default properties required at runtime (path to third-party executables for instance).

Still in the plugin's folder, a site-specific properties file, *site.properties*, is to be used for defining the local configuration of the plugin. Note that this file will be copied when upgrading the plugin.

Backups

Opal assigns a data folder location to the plugin: **OPAL_HOME/data/<plugin-name>** where *plugin-name* is the name defined in the *plugin.properties* file. This folder is then the one to be backed-up.

R Server Installation Guide

Contents of this Guide

Introduction

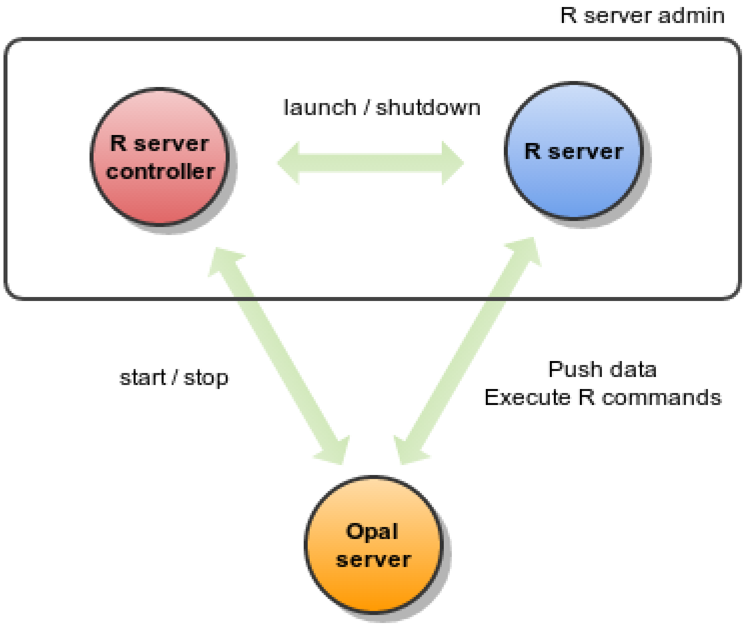
Opal is able to interact with a R server for running statistics analysis and reports. See Opal R and DataSHIELD User Guide and Opal Reporting User Guide.

This guide is about how to install and configure an application called "R Server Admin". This application is made of a R server controller that does the following;

- listen to request for starting / stopping the R server,
- launch / shutdown the R server upon request.

Typical usage is the ability to start / stop a R server from the Opal R Server Administration User Interface. This decoupling of Opal and the R server allows to:

- run the R server on a different host,
- run the R server on behalf of a user having limited rights (in particularly, not having access to Opal server files).



Once the R server is installed and configured, see the Opal R Server Configuration section.

R Server Admin Installation

Software Requirements

Software	Suggested Version	Download Link	Use	Installation/Configuration
Java	>= Java 7.x	http://www.oracle.com/technetwork/java/javase/downloads/index.html	Java runtime environment - needed to run Opal	Java 7 Installation Guide Java 8 Installation Guide Java 8 Ubuntu Installation Guide
R	>= 3.0.x	http://cran.r-project.org	Statistical analysis engine	R Project

Installation of Opal R Server Debian package (recommended)

Opal R Server is available as a Debian package from OBiBa Debian repository. Installing this Debian package will also install R Server Admin, R and Rserve Debian packages. It will also install R packages required for Opal Reporting.

Download Opal R Server Debian package

Or if OBiBa Debian repository was already added to your system:

```
sudo apt-get install opal-rserver
sudo service rserver restart
```

Installation of Opal R Server RPM package (recommended on RPM-based Linux distributions)

Opal R Server is available as a RPM package from OBiBa RPM repository.

Download Opal R Server RPM package

Or if OBiBa RPM repository was already added to your system:

```
sudo yum install opal-rserver
sudo service rserver restart
```

Installation of R Server Admin Zip distribution

R Server Admin is also available as a Zip file.

Download R Server Admin Zip package

Installing Rserve

Rserve is a R package that needs to be installed. This can be done within R by using the `CRAN install command` :

```
install.packages(c('Rserve', 'opal', 'ggplot2', 'opaladdons'),
repos=c('http://cran.rstudio.com/', 'http://cran.obiba.org'),
dependencies=TRUE)
```

Upgrading from R Server Admin 1.2.0

To safely upgrade R Server Admin <= 1.20 from a RPM package only, please remove the current installation as shown in the steps below before installing the new version. Releases following 1.2.0 address a packaging bug jeopardizing data folders upon removal:

```
sudo /etc/init.d/rserver stop
sudo rpm -e rserver-admin --noscripts
```

Please note that the precautionary steps above do not apply to the Debian installation.

R Server Admin Configuration

R Server Admin package has two configuration files: one for the R server controller and one for the R server itself.

R Server Controller Configuration

The file **application.properties** allows the configuration of the R server controller. This one provides REST web services to start/stop a R server.

Property	Description
`server.port`	R server controller port (default is 6312).
`r.exec`	R executable path, required to launch the R server.

R Server configuration

The file **Rserv.conf** allows the configuration of the R server. See the full documentation of this configuration file.

By default the R server has the following configuration:

- connection port is 6311,
- remote connection is disabled,
- no authentication is required.

If the R server is installed on a different machine as the Opal server, you typically will have to:

- enable remote connection,

- enable authentication.

R Server Admin Execution

Launching R Server Admin

> R will create a directory in the R server user home when the first R package is installed: `~/R/<arch>-library/<R-version>`
>
> `**R server needs to be restarted after this directory is created** , so that the R process has this directory in its library look-up paths.`

Service

If the `rserver-admin` package was installed as a system service (either from a Debian package or by providing your own init script), available service commands are described by running:

```
/etc/init.d/rserver help
```

Manually

Manual command is in the `bin` directory of the distribution.

> For security reason you must run R Server Admin on behalf of a user having limited permissions. This is due to the fact that Opal users (that are granted R or Reporting usage) can run arbitrary R code (does not apply to DataSHIELD users).

On a linux system:

```
rserver-admin
```

On a Windows system:

```
rserver-admin.bat
```

Using R Server Admin

R Server Admin is a REST server and therefore can be queried using the curl tool.

```
# R Server Admin requests

# status of the R server
curl localhost:6312/rserver

# start R server (ignored if already started)
curl -X PUT localhost:6312/rserver

# stop R server (ignored if already stopped)
curl -X DELETE localhost:6312/rserver
```

Opal Configuration Guide

Contents of this Guide

Prerequisites

Opal is installed, following the instructions described in Opal Installation Guide.

Opal Configuration Files

Opal has some configuration files that allows fine tuning of your Opal server. See Opal Configuration Files.

Users Directories Configuration

Opal comes with several user directories. Others can be added. See User Directories Configuration.

Reverse Proxy Configuration

Opal server can be accessed through a reverse proxy server.

Apache

Example of Apache directives that:

- redirects http connection on port 80 to https connection on port 443,
- defines organization's specific certificate and private key.

```
<VirtualHost *:80>
    ServerName opal.your-organization.org
    ProxyRequests Off
    ProxyPreserveHost On
    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>
    RewriteEngine on
    ReWriteCond %{SERVER_PORT} !^443$
    RewriteRule ^/(.*) https://opal.your-organization.org:443/$1 [NC,R,L]
</VirtualHost>
<VirtualHost *:443>
    ServerName opal.your-organization.org
    SSLProxyEngine on
    SSLEngine on
    SSLProtocol All -SSLv2 -SSLv3
    SSLHonorCipherOrder on
    # Prefer PFS, allow TLS, avoid SSL, for IE8 on XP still allow 3DES
    SSLCipherSuite "EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM EECDH+ECDSA+SHA384
EECDH+ECDSA+SHA256 EECDH+aRSA+SHA384 EECDH+aRSA+SHA256 EECDH+AESG CM EECDH
EDH+AESGCM EDH+aRSA HIGH !MEDIUM !LOW !aNULL !eNULL !LOW !RC4 !MD5 !EXP
!PSK !SRP !DSS"
    # Prevent CRIME/BREACH compression attacks
    SSLCompression Off
    SSLCertificateFile /etc/apache2/ssl/cert/your-organization.org.crt
    SSLCertificateKeyFile
/etc/apache2/ssl/private/your-organization.org.key
    ProxyRequests Off
    ProxyPreserveHost On
    ProxyPass / https://localhost:8443/
    ProxyPassReverse / https://localhost:8443/
</VirtualHost>
```

Opal Monitoring

Opal can be monitored using New Relic application. New Relic is a managed service (SaaS) which collects and aggregates performance metrics of your Opal application.

Opal is preconfigured for New Relic but **monitoring is disabled by default.**

To enable New Relic monitoring:

- create a free account to get a license key. This key binds your Opal's data to your account in the New Relic service
- copy this license key to **license_key** property in **OPAL_HOME/conf/newrelic.yml** file (line 15)
- if you plan to monitor multiple Opal instances, set **app_name** property (line 30)
- restart Opal

New Relic monitoring is free for a 24 hour data retention.

Opal Administration User Interface

Part of Opal administration can be done using the web interface. See details about how to configure Opal system in Administrating Opal.

Opal Configuration Files

Overview

Summary

The following configuration files are available in the **`OPAL_HOME/conf`** directory.

File	Description
`shiro.ini`	The user authentication configuration file.
`opal-config.properties`	The main configuration file to be edited before starting Opal.
`newrelic.yml`	The New Relic configuration file for advanced server monitoring.

The following configuration files are available in the **`OPAL_HOME/data`** directory.

File	Description
`opal-config.xml`	The advanced configuration file: the file system is to be edited before starting Opal, other entries can be configured using the administration web interface.

Opal Main Configuration File

The file **`OPAL_HOME/conf/opal-config.properties`** is to be edited to match your server needs.

Configuration Property	Description
`org.obiba.opal.keys.*`	Identifiers Table Configuration
`org.obiba.opal.ssh.port`	SSH Server Configuration
`org.obiba.opal.http*`	HTTP Server Configuration
`org.obiba.opal.smtp.*`	SMTP Server Configuration
`org.obiba.opal.Rserve.*`	R Server Configuration
`org.obiba.realm.*`	Agate Server Configuration

Identifiers Table Configuration

The identifiers are stored in a Datasource Table which names can be specified.

Configuration Property	Description
`org.obiba.opal.keys.tableReference`	Fully-qualified name of the identifiers table
`org.obiba.opal.keys.entityType`	Type of entities to store in the identifiers table.

SSH Server Configuration

Opal is accessible using SSH clients: command line user interface and SFTP are available through SSH connections. The SSH connection port can be configured.

Configuration Property	Description
`org.obiba.opal.ssh.port`	The port to use for listening for SSH connections. Default value is 8022.

HTTP Server Configuration

Opal web services and web application user interface can be accessed through HTTP or secured HTTP requests. The HTTP(S) connection ports can be configured.

Configuration Property	Description
`org.obiba.opal.http.port`	The port to use for listening for HTTP connections. Default value is 8080, -1 to disable.
`org.obiba.opal.https.port`	The port to use for listening for HTTPS connections. Default value is 8443, -1 to disable.
`org.obiba.opal.ajp.port`	The port to use for the Apache JServ Protocol. Default is -1 (disabled).
`org.obiba.opal.maxIdleTime`	the maximum time a single read/write HTTP operation can take in millis (default is 30000). See idleTimeout Jetty configuration.
`org.obiba.opal.ssl.excludedProtocols`	Specify the SSL/TLS protocols to be excluded. Usually *SSLv3* will be excluded. Use commas for separating multiple protocol names. Default is no protocol is excluded (for legacy reason). See JSSE Provider documentation.
`org.obiba.opal.ssl.includedCipherSuites`	Specify which Cipher Suites to be included. Use commas for separating multiple cipher suites names. Default is all that is available. See JSSE Provider documentation.

The HTTPS server requires a certificate. If none can be found Opal creates a default one to ensure that HTTPS is always available. It should be configured afterward, following the procedure described in HTTPS Configuration.

SMTP Server Configuration

Opal is able to send emails to notify that a rapport has been produced. To allow this, it is required to configuration to a SMTP server.

Configuration Property	Description
`org.obiba.opal.smtp.host`	The SMTP server host name.
`org.obiba.opal.smtp.port`	The SMTP server port number.
`org.obiba.opal.smtp.from`	The "From" email address when sending emails.
`org.obiba.opal.smtp.auth`	A flag to indicated if authentication against SMTP server is required. Allowed values are: `true/false`. Default is false (usually not required when server is in the same intranet).

`org.obiba.opal.smtp.username`	The SMTP user name to be authenticate (if authentication is activated).
`org.obiba.opal.smtp.password`	The SMTP user password (if authentication is activated).

R Server Configuration

Opal is able to perform R queries by talking with a running Rserve. **Opal does not provide R and Rserve**: see R Server Installation Guide. Rserve version must be 0.6+. The properties for connecting to Rserve are the following:

Configuration Property	Description
`org.obiba.rserver.port`	Port number of the R server controller (allows Opal to start/stop the R server).
`org.obiba.opal.Rserve.host`	Hostname of the Rserve daemon (default is blank, i.e. the one defined by Rserve (localhost))
`org.obiba.opal.Rserve.port`	TCP port to connect to (default is blank, i.e. the one defined by Rserve (6311))
`org.obiba.opal.Rserve.username`	Username to use for login-in to Rserve (none by default)
`org.obiba.opal.Rserve.password`	Password to use for login-in to Rserve (none by default)
`org.obiba.opal.Rserve.encoding`	Character encoding for strings (default is utf8)
`org.obiba.opal.r.sessionTimeout`	Time in minutes after which an active R session will be automatically terminated (default is 4 hours).

Agate Server Configuration

Opal user lookup can include the Agate's user realm. Default configuration enables connection to a Agate server.

Configuration Property	Description
`org.obiba.realm.url`	Address to connect to Agate server. Default is https://localhost:8444. To disable Agate connection, specify an empty value for this property.
`org.obiba.realm.service.name`	Application name of this Opal instance in Agate. Default is *opal*.
`org.obiba.realm.service.key`	Application key of this Opal instance in Agate. Default is *changeit*.

Taxonomies Configuration

Opal is able to read taxonomies at start-up from provided URIs. Defaults are the taxonomies defined by Maelstrom Research.

Configuration Property	Description
`org.obiba.opal.taxonomies`	Comma separated list of URIs to taxonomy files in YAML format. Note that *file* URI schema is supported (allows to read locally defined taxonomy).

Plugins Configuration

Configuration Property	Description
`org.obiba.opal.plugins.site`	The URL to the plugins repository (default is https://plugins.obiba.org/stable). A plugin repository is not just a list of files, meta-data information about plugins are expected to be provided by a plugins.json file.

Miscelaneous Configuration

Advanced settings.

Configuration Property	Description
`org.obiba.opal.ssl.excludedProtocols`	SSL/TLS (comma separated) protocols that HTTPS server must not reply to. Typical configura is to not exclude any of the SSL/TLS protocols.
`org.obiba.opal.maxFormContentSize`	Maximum body size of a HTTP(S) form post request. Default value is "`200000`" bytes.
`org.obiba.opal.ws.messageSizeLimit`	Limit of the Protobuf message size. Default value is "`524288000`" bytes (500MB).
`org.obiba.magma.entityIdNames`	Specify the column name per entity type to be used for the entity identifier when exporting dat If empty for the considered entity type, the default column name will apply. The format to be u list, for instance: `org.obiba.magma.entityIdNames=Participant=Idepic,E`
`org.obiba.magma.entityIdName`	Specify the default column name to be used for the entity identifier when exporting data to a fi empty, this name depends on the file format.

Opal Advanced Configuration File

The file **`OPAL_HOME/data/opal-config.xml`** can be edited to match some of your server needs.

File System Root

Opal offers a "file system" in which users may manipulate files without having a user defined in the OS running Opal. That is, all interactions with the underlying file-system go through a unique system-user: the one that runs the Opal server.

The Opal file system root is set by default to be `OPAL_HOME/fs`. To change it, modify the following statement:

opal-config.xml

```
<!-- Windows example -->
  <fileSystemRoot>C:/opal-filesystem</fileSystemRoot>
```

Several types of file root names are recognized:

- Absolute URI. These must start with a scheme, such as '`file:`', followed by a scheme dependent file name. For example:
  - `file:/c:/dir/somedir`
- Absolute local file name. For example, `/home/someuser/somedir` or `c:\dir\somedir`. Elements in the name can be separated using any of the following characters: /, \, or the native file separator character. For example, the following file names are the same:
  - `c:\dir\somedir`
  - `c:/dir/somedir`

User Directories Configuration

Overview

Summary

The security framework that is used by Opal for authentication, authorization etc. is Shiro. Configuring Shiro for Opal is done via the file **`OPAL_HOME/conf/shiro.ini`**. What ever changes are done in this file requires opal to be restart to be effective.

More information about how to configure Shiro using the INI file can be found here.

Default configuration is a static user `'administrator'` with password `'password'` (or the one provided while installing Opal Debian package).

Remember to change default administrator password if you did not installed Opal with the Debian package!

User Directories

By default Opal has several built-in user directories:

- a file-based user directory (`shiro.ini` file),
- an internal user directory for users who can login with password,
- an internal user directory for users who can login with a certificate.

Additional user directories can be added for matching your organisation needs.

In the world of Shiro, a user directory is called a `realm`.

File Based User Directory

The file-based user directory configuration file **OPAL_HOME/conf/shiro.ini**.

> It is not recommended to use this file-based user directory. It is mainly dedicated to define a default system super-user. Refer to the Users and Groups Administration documentation for adding new users.

For a better security, user passwords are encrypted with a one way hash such as sha256.

The example `shiro.ini` file below demonstrates how encryption is configured.

conf/shiro.ini

```
# =======================
# Shiro INI configuration
# =======================

[main]
# Objects and their properties are defined here,
# Such as the securityManager, Realms and anything else needed to build the
SecurityManager

[users]
# The 'users' section is for simple deployments
# when you only need a small number of statically-defined set of User
accounts.
#
# Password here must be encrypted!
# Use shiro-hasher tools to encrypt your passwords:
#   UNIX:
#     cd /var/lib/opal/tools && ./shiro-hasher -p
#   WINDOWS:
#     cd <OPAL_DIST_HOME>/tools && shiro-hasher.bat -p
#
# Format is:
# username=password[,role]*
administrator =
$shiro1$SHA-256$500000$dxucP0IgyO99rdL0Ltj1Qg==$qssS60kTC7TqE61/JFrX/OEk0j
sZbYXjiGhR7/t+XNY=,admin

[roles]
# The 'roles' section is for simple deployments
# when you only need a small number of statically-defined roles.
# Format is:
# role=permission[,permission]*
admin = *
```

Passwords must be encrypted using **shiro-hasher** tools (included in Opal tools directory):

Password encryption for Linux

```
cd /var/lib/opal/tools
./shiro-hasher -p
```

Password encryption for Windows

```
c:\>cd <OPAL_DIST_HOME>/tools
c:\>shiro-hasher.bat -p
```

LDAP and Active Directory Authentication

Opal can authenticate users by using an existing LDAP or Active Directory server. This is done by adding the proper configuration section in the `shiro.ini` file:

LDAP Configuration

```
[main]
ldapRealm = org.apache.shiro.realm.ldap.JndiLdapRealm
ldapRealm.contextFactory.url = ldap://ldap.hostname.or.ip:389
ldapRealm.userDnTemplate = uid={0},ou=users,dc=mycompany,dc=com
```

The `userDnTemplate` should be modified to match your LDAP schema. The `{0}` will be replaced by the username provided at login. Authentication will use the user's credentials to try to bind to LDAP; if binding succeeds, the credentials are considered valid and authentication will succeed.

> There is currently no support to extract a user's groups from LDAP. This will be added in a future release.

With Active Directory you can specify a mapping between AD groups and roles in Shiro. Example configuration for Active Directory authentication:

Active Directory Configuration

```
[main]
adRealm = org.apache.shiro.realm.activedirectory.ActiveDirectoryRealm
adRealm.url = ldap://ad.hostname.or.ip:389
adRealm.systemUsername = usernameToConnectToAD
adRealm.systemPassword = passwordToConnectToAD
adRealm.searchBase = "CN=Users,DC=myorg"
adRealm.groupRolesMap = "CN=shiroGroup,CN=Users,DC=myorg":"myrole"
#adRealm.principalSuffix =
```

Atlassian Crowd User Directory

Opal can authenticate users by using an Crowd.

First, you need to enable Crowd client by uncommenting the following in **OPAL_HOME/conf/custom-context.xml**:

OPAL_HOME/conf/custom-context.xml

```
<import resource="file:${OPAL_HOME}/conf/crowd/crowd-context.xml" />
```

Then configure your Crowd client in **OPAL_HOME/conf/crowd/crowd.properties**:

OPAL_HOME/conf/crowd/crowd.properties

```
application.name=application
application.password=password
application.login.url=http://crowd.example.com/crowd/console/
crowd.server.url=http://crowd.example.com/crowd/services/
crowd.base.url=http://crowd.example.com/crowd/
session.isauthenticated=session.isauthenticated
session.tokenkey=session.tokenkey
session.validationinterval=2
session.lastvalidation=session.lastvalidation
```

Other Settings

Session Timeout

Shiro's default session timeout is 1800s (half an hour). The session timeout can be set explicitly in the shiro.ini file, in the `[main]` section:

conf/shiro.ini

```
# =======================
# Shiro INI configuration
# =======================

[main]
# Objects and their properties are defined here,
# Such as the securityManager, Realms and anything else needed to build the
SecurityManager
# 3,600,000 milliseconds = 1 hour
securityManager.sessionManager.globalSessionTimeout = 3600000

# ...
```

The session timeout is in *milliseconds* and allowed values are:

- a negative value means sessions never expire.
- a non-negative value (0 or greater) means session timeout will occur as expected.

Opal Web Application User Guide

Contents of this Guide

Summary

This guide provides a description of the web interface.

To access Opal with a web browser the following urls may be used (port numbers may be different depending on HTTP Server Configuration):

- `http://<servername>:8080` will provide a connection without encryption,
- `https://<servername>:8443` will provide a connection secured with ssl.

Prerequisites

On server side: a properly configured Opal distribution. See Opal Installation Guide and Opal Configuration Guide for details about it.

On client side: a Browser with connection information (host, port). Firefox or Chrome are recommended (see details in Installation Guide about Client Software Requirements).

Using Web Interface

- Browse Datasources: browse tables and variables, create views, import/export data and dictionaries.
- Manage Participant Identifiers: create and map identifiers of participants in different units, manage encryption keys.
- Run Reports: design reports with BIRT over Opal data and schedule their execution.
- Manage Files: browse, upload and download files on the server.
- Manage Jobs: list and manage jobs that were launched on the server.
- Administer Opal: modify R-DataShield setup and permissions.

Post Installation Set-up

The first time you connect to Opal (as an administrator), you will automatically be redirected to the databases administration page: Opal is made of two distinct databases: one for the holding the participants identifiers and another one for holding the variable catalogue and participant data.
Opal requires to have one identifiers database and at least one data database registered to be fully functional.

Defining a different database user for each of these databases ensures the participant data privacy. See Identifiers Mappings section for more details.

If you are using a Opal server deployed from a Amazon Machine Image or a Vagrant Virtual Machine, this set-up has already been done for you.

Refer to Database Administration section to choose and configure your databases.

Working with Projects

Contents of this Guide

Summary

This guide provides a description of the web interface for managing projects.

A project is the workspace for managing and transforming your data. It is required to create a project before starting to work with variables and data.

A project is made of:

- A database which provides services for the persistence of the variable dictionary and of the data: see Datasource Types and Project Tables documentation,
- A repository of views (logical tables with derived variables) with their change history,
- A repository of report templates (and generated reports): see Project Reports documentation,
- A folder in the Opal file system: see Project Files documentation,
- Progress of the tasks performed in the context of the project can be followed: see Project Tasks documentation,
- The permissions that applies to the tables, the variables, the reports and the project itself: see Project Permissions documentation,
- A repository of encryption keys: see Project Administration documentation.

See also Variables organization for a general description of how the data are described in Opal.

Operations

Add Project

A project is where Opal will store the tables (i.e. variables and data). When creating a project, the database that can be chosen must be one of the databases described in the Databases Administration section (note that only databases which usage is *Storage* can be used to store the data of a project).

If no database is selected (*None* option) only views could be added to the project: no data can be imported and only logical tables (i.e. views) can be added. Note that it is still possible to declare a database associated to this project afterwards.

Import Data

Go to a project tables section and start Import Data procedure.

Export Data

Go to a project tables section and start Export Data procedure with all tables of the project pre-selected. To export the data of a subset of tables, select some tables before executing this operation.

Table Details

From a project the list of tables, the details of the selected table are displayed. See complete description of content and operations in table details documentation.

Variable Details

From the list of variables in table details, the details of the selected variable are displayed. See complete description of content and operations in variable details documentation.

Datasource Types

Overview

Introduction

Datasources are the entry point in Opal for accessing to Variables and Data. Datasources can be of different kinds, some being more suitable for different purposes (variables import, data import and export, permanent storage).

Datasource type	Variables Import	Variables and Data Import	Variables and Data Export	Storage
CSV Datasource		✓	✓	
Opal Archive Datasource		✓	✓	
SPSS Datasource	✓	✓		

SPSS R Datasource		✓	✓	
SAS R Datasource		✓	✓	
Stata R Datasource		✓	✓	
Excel Datasource	✓			
Opal SQL Datasource				✓
Tabular SQL Datasource		✓	✓	✓
Limesurvey Datasource		✓		
Opal Datasource		✓		
MongoDB Datasource				✓

File Based Datasources

File based datasources are convenient for import and export operations.

CSV Datasource

CSV datasource will expect the file to use a "delimiter separated values" format (default delimiter being comma). The first column will represent the entity identifiers and the subsequent column names will identify variables. Each row of the file (except the first row) are the values for one entity. The entity identifier must be unique: there cannot be two rows starting with the same identifier.

Due to the nature of the CSV format, the data dictionary is limited to the variable names (i.e. the name of the columns). A CSV file can be imported as-is but the variables will be considered as being of text type only. When importing CSV data, if the destination table already exists, Opal will consider that the data dictionary of the CSV file is the one of the destination table. Then before importing CSV data it is recommended to prepare the destination table variables first: see Add Table section.

Example

The following data dictionary is used in this example:

- Var1: `text` value type
- Var2: `integer` value type
- Var3: `text` value type, repeatable (i.e. each value is a sequence of value)

The data to be represented in CSV are for instance:

ID	Var1	Var2	Var3
123	This is a value	1	"Value 1","Value 2"
234		2	x,y
345	This is a multi-line value		a

The CSV file uses the options:

- the separator character: `,`
- the quote character: `"`

data.csv

```
ID,Var1,Var2,Var3
123,"This is a value",1,"""Value 1"",""Value 2"""
234,,2,"x,y"
345,"This is a
multi-line value",,a
```

For more information about CSV format:

- Comma separated values
- Delimiter separated values
- RFC4180

Opal Archive Datasource

Opal Archive datasource is a fully featured file-based datasource. This datasource comes as a `.zip` file (that can be optionally encrypted) containing a folder for each table having: the full data dictionary in a XML file, a XML data file per entity. This is the file format used when exporting data from Onyx.

SPSS Datasource

An SPSS datasource is a read-only datasource. The SPSS source file must be a valid non-compressed binary file with a `.sav` extension. In Opal an SPSS file represents a table and its variables are used as the table's data dictionary. An Opal compatible SPSS file must have its first variable represent the identifiers. If this is not the case, before a file import, the identifier variable must be moved to the first position of the SPSS variable sheet.

The following SPSS variable attributes are imported to the data dictionary:

- width
- decimals
- measure
- shortname
- format (F9.2, ADate10, etc)

In addition, variable categories and missing values are also imported and converted to their Opal counterparts

> Currently, Opal does not handle missing values with large intervals (-9999..9999). Until a more robust solution is implemented, try to keep the intervals small or discrete.

Excel Datasource

Opal supports both Excel 97 and Excel 2007 formats. Excel format limitations are:

Extension	Format used	Limits
`.xls`	Excel 97	256 columns and 64K lines
`.xlsx`	Excel 2007	16K columns and 1M lines

R Based Datasources

R based datasources are datasources that are using R server to extract/write data in a given format. The supported formats are the ones defined in the haven R package (package which is expected to be installed on the R server). Note that this is still an experimental feature: value type mappings with R could change in a future release and some limitations of the haven package may apply.

SPSS R Datasource

The expected/produced file extension is `.sav`.

SAS R Datasource

The expected/produced file extension is `.sas7bdat`. If when importing, a file exists with same base name in the same parent folder and with extension `.sas7bcat`, it will be automatically used as the catalog file.

Stata R Datasource

The expected/produced file extension is `.dta`.

SQL Based Datasources

SQL based datasources are convenient for variables and data storage. With some limitations, this type of datasource can be used for import and export.

Opal SQL Datasource

Opal SQL is the most versatile datasource type with MongoDB datasource. The underlying SQL database schema is a EAV which allows to store an unlimited number of variables.

For more information about this datasource see Opal SQL Schema documentation.

Tabular SQL Datasource

Tabular SQL datasources are suitable for datasets with a (relatively) small number of variables. Data copied into Tabular SQL datasource are stored in classical SQL tables, i.e. one row per entity and one variable per column. Check SQL database vendor specifications to know the number of columns (i.e. variables) that can be defined for a table: see for instance MySQL Table Column-Count and Row-Size Limits. Comprehensive meta-data for each column field can be optionally stored in separated tables. Opal is able to increment copies into Tabular SQL datasources if update timestamp column is given.

For more information about this datasource see Tabular SQL Schema documentation.

Document Oriented Datasources

NoSQL document oriented datasources are convenient alternative to SQL based datasources. It allows to store an unlimited number of variables.

MongoDB Datasource

MongoDB is the most versatile datasource type with Opal SQL datasource.

Other Server Based Datasources

Server based datasources are convenient for import operations, from a data collection application usually.

Limesurvey Datasource

Limesurvey datasource is able to extract, from a Limesurvey SQL database, one table per survey with its fully described data dictionary. The data that will be imported are the interviews that are completed.

Opal Datasource

Opal datasource allows one Opal server to connect to a remote Opal server. This can be useful when syncing datasources in different Opal instances.

Project Tables

Overview

- Add Table
- Add/Update Tables from Dictionary
- Add View
- Remove

Summary

Tables are give access to the project data along with their description. A table can be a raw table (i.e. with data persisted in the project's database) or a logical table (also called view) which is a set of derived variables (data are computed on-demand).

For more details see also:

- Table Details
- Variable Details
- Querying for Variables

Dictionary

List of the tables in the project with summary information for each of them:

- entity type
- number of variables
- number of entities

Operations

Download Dictionary

The whole project data dictionary can be download as an Excel file. This file is compatible with the operations of Add/Update Tables and Add View. When no tables are selected, the downloaded dictionary contains the definition of all tables. When some tables are selected, the dictionary contains only the definition of selected tables.

Import Data

When importing data, Opal relies on the concept of datasource. This allows Opal to abstract the data importation process from the source datasource to the destination datasource regardless of their underlying implementations (file, SQL database etc.).

The importation process follows several steps:

- Data format selection: file-based (CSV, Opal Archive, SPSS) or server-based (SQL, Limesurvey, Opal).
- Data format specific options,
- Incremental options,
- Identifiers mapping options,
- Data dictionary update review and table to import selection,
- Data to import review,
- Archiving options when dealing with a file-based datasource.

Some import options can be described as follow:

Option	Description
Incremental	Opal is able to detect new or updated data, relying on entity identifier and some timestamps. By default the data import is not incremental, i.e. already existing data will be overridden.
Limit	A maximum number of data rows to be imported. Combined with the *Incremental* option, this allow to import small chunks of data at a time.

Identifiers Mapping	If an identifiers mapping is selected, each participant identifier encountered in the imported datasource is expected to be one of the identifiers registered for this mapping. Depending on the identifiers mapping strategy the import could fail: • Each identifiers must be mapped prior importation (default): the import will fail if an imported identifier does not have a corresponding system identifier for the selected mapping, • Ignore unknown identifiers at import: only data with identifier having a corresponding system identifier in the selected mapping will be imported, • Generate a system identifier for each unknown imported identifiers: a system identifier will be generated for each unknown imported identifier. If no identifiers mapping is selected, the participant identifiers are imported as-is. Unknown participant identifiers will be automatically added in Opal. See Identifiers Mappings and Identifiers Mapping Administration documentations for more details.

Export Data

Selected tables (or currently viewed table) can be exported. The exportation process offers several options:

- Data format selection: file-based (CSV, Opal Archive) or server-based (SQL),
- Data format specific options: destination folder or export database name,
- Values filter options: available when a filter has been applied on the table's values,
- Identifiers mapping options:
  - If a identifiers mapping is selected, each entity to be exported must have a mapped identifier. Otherwise the export will fail.
  - If no identifiers mapping is selected the data are exported with system identifiers.

See Identifiers Mappings and Identifiers Mapping Administration documentations for more details.

Copy Data

Selected tables (or all tables) can be copied into another project or in the same project but with a different name (table renaming is available only when one table is selected for copy).

Add Table

Adds a table to the project. Each table must have a unique name and an entity type.

Add/Update Tables from Dictionary

This operation follows a step-by-step procedure:

1. Specify a data dictionary file. Currently supported formats are Excel, SPSS and XML view:
   - An Excel file describes the data dictionary. The Excel template file is available to download from this step.
   - The data dictionary of a SPSS file will be transformed as a table,
   - Variables from a XML view file will also be turned into a table (not a view).
2. Select which tables will be added or updated (in case a table with same name exists). You can check the details of each table variables that will be added/updated. In case of the table that is updated is a view, the variables of this view will be added/updated.

Add View

This operation follows a step-by-step procedure:

1. Specify the view name and the data dictionary (optional). The data dictionary can be provided as an xml file (this can be obtained from an existing view by selecting "Download View XML") or an Excel file (see Excel file template). If a view with same name already exists, confirmation for overriding it is required. If a plain table already exists with same name, the operation is not allowed.
2. Specify which tables this view refers to (required).

Derived variables algorithms are expressed using Magma Javascript API.

Remove

Removes the selected tables/views from the project and deletes its data.

Table Details

Overview

Introduction

A table can be a raw table or a view.

Dictionary

List of the variables in the table with summary information for each of them:

- label (mapped on `label` variable atribute if this one is defined)
- value type
- unit

Summary

The data of the table can be indexed in Opal's internal search engine. Indexing the values allows:

- to have pre-computed variable summaries,
- to filter the table entities by their values (and subsequently copy/export a subset of the data).

Values

The values of the table can be seen in this section. The view port on the values is limited by a number of rows and a number of visible variables (display options allow to modify these numbers). Right and left arrows in table header (with the variable names) allows to move the variables view port.

When the table has been indexed (see Summary section), the rows can be filtered by variable criteria. The resulting subset of data can be exported/copied using the usual Export/Copy procedure.

Permissions

Specify the access rights to a particular table and its content.

View dictionary and summaries Permission

Allow the user to see data dictionary with variable data summaries. Does not allow values querying. It induces the read-only access to the parent datasource.

View values Permission

Allow the user to see the table's data: values querying services are available. Automatically grants the View dictionary and summaries Permission
.

Edit dictionary Permission

Applies only to plain tables (i.e. not views).

Allow edition of the table's data dictionary. Automatically grants the View dictionary and summaries Permission.

Edit with summaries Permission

Applies only to tables that are views.

Allow edition of the view's data dictionary, i.e. the edition of the derived variables algorithms. Automatically grants the View dictionary and summaries Permission. This permission does not grant access to individual-level data.

Edit with values

Applies only to tables that are views.

Allow edition of the view's data dictionary, i.e. the edition of the derived variables algorithms. Automatically grants the View dictionary and summaries Permission. This permission grants access to individual-level data.

Administrate Table Permission

Allow all operations on the table/view (including removing it).

Operations

Add Variables to View

This operation consist of making a derived variable for each of the selected variables (see #Variable Selection) and adding them to a view (either an existing one or one that would be created for that purpose). Options are:

- derived variable name can be changed (default is the original variable name),
- categorical variables can be recoded (i.e. category names are turned to a numerical value).

If a derived variable with the same name already exists in the destination view, this derived variable will be overwritten with the new definition. Else a new derived variable is added.

Export Variable Dictionary

The table/view data dictionary can be download as an Excel file. This file is compatible with the operations of Add/Update Tables and Add View.

Export Data

Start Export Data procedure with the table pre-selected.

Copy Data

The table can be copied into another project or in the same project but with a different name.

Remove Table

This operation deletes the data and the data dictionary associated with the table. This cannot be undone.

Variable Selection

Variable can be selected to perform batch operations:

- *Add variable to view*: make an identity derived variable, added to a view.
- *Apply attribute*: apply a custom attribute or a taxonomy term.
- *Remove attributes*: remove variable attributes by specifying namespace (optional) and name, or taxonomy and vocabulary.
- *Remove*: variable and associated data will be removed.

View Specific Operations

Download View XML

Only available if the table is a view.

Edit View

Edit the view properties, i.e. its name and the table references: these tables can be ordered and can be flagged as being *inner*. A *inner* table means that the entities of this table do not contribute to the entities of the view (similar to a SQL inner join). A typical use case is when data collected by the study are joined with data from a governmental database: if one would like to restrict the participants of the resulting view to the ones that of the study, the governmental table would be joined to the view as a *inner* table.

Remove View

This operation will only remove the logical description of the view. It will not affect the referred data.

Entity Filter

A script can be defined to restrict the view entities to the ones matching some criterai (for instance, all women older than 50 years). This script must return a logical value: *true*, the entity is kept, *false* (or *null*), it is excluded.

Variable Search

Variables of a table can be searched as specified in Querying for Variables. Selecting the suggested name goes to the corresponding variable details.

Variables List Filtering

The list of the variables can be filtered the same way the variables can be searched: see Querying for Variables. On `ENTER` key pressed, the list is refreshed with all the variables matching the criteria.

Variable Details

Overview

Introduction

A variable describes the data.

Operations

Add variable to view

This operation adds or updates a derived variable in a view for each selected variable.

Categorize this variable to another

This operation adds or overwrites a variable in a view and allows to recode its values.

Categorize another variable to this

This operation maps another variable's values to the current variable categories.

Custom derivation

Derive a variable by editing its derivation script manually.

Remove

Removes the variable from the table.

Properties

This section displays the properties of the variable:

- Name
- Entity type
- Value type
- Repeatable
- Unit
- Referenced Entity Type
- Mime type
- Occurrence Group
- Index

Categories

Some variables can have categories defined. The list of categories is displayed with a summary information:

- label (mapped on `label` category attribute if this one is defined),
- missing if the category indicates a missing answer.

Edit Categories

This operations allows the addition, edition and deletion of variable's categories. Categories can also be removed or reordered by selecting one or multiple categories.

Attributes

Some variables can have attributes defined. The list of attributes is displayed with a full information:

- namespace
- language
- value

Add Attribute

This operation adds a new attribute. The combination of namespace and name must be unique.

Edit

To assign the attribute to another namespace, change its name or setting its values. When editing multiple attributes only the namespace can be modified.

Remove

Removes the attributes.

Summary

Statistical summary of the variable:

- variables with categories:
  - frequency plot
- variables without categories:
  - histogram
  - normal probability plot
  - summary data: N, Min, Max, Mean, Median etc.
  - Frequencies of missing and non-missing values.

Script

Derived variables (i.e. when the table is a view) are persisted in Opal's embedded Version Control System which tracks all changes to a script over time. One practical use case is revising the history of changes and if necessary revert the script to a previous revision.

Script History Revisions

Each time a script is edited a new history revision is created or 'committed' to Opal's VCS.

Commit Differences

Commit revisions are organized in a descending order, i.e., the latest commit at the top of the history stack. A simple 'diff' compares the changes between two immediate commits. Opal also offers a comparison between any revisions to the current revision.

Reverting Changes

By editing and saving an older revision, a script content is reverted to its previous version. This operation is tracked as a new revision.

Review Commit Differences

The commit differences are ordered by the oldest changes first (denoted in red) followed by the latest changes (denoted in green).

Values

Values can be displayed for a specific identifier or can be filtered to match to certain criterias.

Permissions

Specify the access rights to a particular variable and its content.

View with summary Permission

Allow the user to see the variable details with its data summary. Does not allow values querying. It induces the read-only access to the parent table and datasource.

Querying for Variables

Overview

- Introduction
- How to Search
    - Search Fields
    - Examples

Introduction

The data dictionary (i.e. the variables) can be searched for. Each variable properties, Categories and Attributes have been indexed in Opal search engine. Each time a table is updated, its data dictionary is re-indexed automatically, ensuring an up-to-date variable search service.

Note for the administrators: variables of a table are always indexed (with a latency of 1 minute), whereas the indexing of the table values can be scheduled.

How to Search

When navigating in the data dictionary (datasources, tables, variables), the search box is pre-filled with the current context (datasource or table currently visited). Start typing a word that is looked for and the 10 first most relevant variables will be suggested. Selecting one of them will display it (see Variable Details).

By default the fields that are searched for are:

- `name`,
- `label` attribute (any language),
- `description` attribute (any language),
- `maelstrom` attributes (any language).

Other fields can be searched for by explicitly defining a search term using the pattern: `<field>:<value>`. The search terms can be combined using logical operators `AND` and `OR` (uppercase is required for the operator). The default logical operator is `AND`. Wildcard character `*` can be used on the value. See #Examples.

Search Fields

The search fields corresponding to the variable properties are:

- `datasource`
- `table`
- `name`
- `fullName`
- `entityType`
- `valueType`
- `occurrenceGroup`
- `repeatable`
- `unit`
- `mimetype`
- `referencedEntityType`
- `category`
- `nature`

The search fields corresponding to the variable Attributes are defined by the pattern: `<namespace>-<name>-<locale>` (`namespace` and `locale` are not always defined).

Categories attributes follow the same pattern, prefixed by `category`: `category-<namespace>-<name>-<locale>`.

The nature of the variable can be : CATEGORICAL, CONTINUOUS, TEMPORAL or UNDETERMINED.

Examples

Search for variables having words starting with "smok" (for instance "smoke", "smoked", "smoking" (case insensitive)) in their name or label or description:

```
smok
```

You can provide a more accurate query by specifying the field name that is searched:

```
name:Measure.RES_FVC
```

Criteria can be combined:

```
Measure.RES_ valueType:binary
```

Default operator is AND, but OR and NOT operators with parenthesis can also be specified:

```
RES_F AND NOT (FEV OR FEF)
```

Search for variables of numerical value type (`integer` or `decimal`):

```
valueType:integer OR valueType:decimal
```

Search for variables with repeatable values in kilograms:

```
unit:kg repeatable:true
```

Search for variables having category with name starting with Y:

```
category:Y*
```

Search for a chunk of phrase in English `label` attribute:

```
label-en:"don't know"
```

Search for variables having a category with some words in their English `label`:

```
category-label-en:yes
```

Project Genotypes

Overview

Summary

When a VCF Store plugin has been activated in the Opal system, genotypes in Variant Call Format (VCF or BCF) can be stored within a project. In order to have this functionality operational and because different VCF Store plugins could be available, the project administrator must specify which VCF Store Service plugin is to be used.

Once a project is set-up for handling genotype data, VCF/BCF files can be imported, exported and their samples can be associated to participant IDs.

VCF Store

The purpose of the VCF store is:

- to be a repository of VCF/BCF files within a project,
- to provide basic statistics and summary information on these VCF/BCF files,
- to allow association between genotypes and phenotypes through a mapping of samples (observed in the the VCF files) and participants (having data in the Project Tables) identifiers,
- to identify the case-control samples observed in the VCF files,
- to define different levels of permissions to access to the VCF files.

Variant Call Format

The VCF store supports both VCF flavors: the original VCF (textual) and the binary BCF. The VCF/BCF file is always stored in its compressed form.

Sample-Participant Mapping Table

A table can be provided to enhance the description of the content of the VCF files. The requirements for this table are:

- this table must have the entity type `Sample` (because the data are about the samples),
- the sample IDs are expected to be the ones of the observed sample IDs in the VCF files but it is not an absolute requirement: there could be unknown sample IDs and some observed could not being described,
- a variable will provide, for a given sample, the participant ID (it is recommended to set the *referenced entity type* property of this table to be `Participant`),
- several samples can be associated to the same participant,
- a variable will specify the role of each sample: either an ordinary *sample* or a *control* (it is recommended to define these two categories in this variable).

An example of sample-participant mapping table could be:

ID	ParticipantID	Role
1	00001	sample
2	00002	sample
3	00002	sample
4		control
5		

Once a sample-participant mapping has been defined, the VCF Store will report:

- for the whole VCF store summary, the count of:
    - unique observed samples,
    - unique participants,
    - unique identified samples,
    - unique controls.
- for each VCF file, the count of:
    - observed samples,
    - associated participants,
    - identified samples (i.e. the count of samples associated to a participant); having more identified samples than participants means that some participants have several samples,
    - case-controls; this is not exclusive of being associated to a participant,
    - variants,
    - genotypes.

Permissions

Specify the access rights to the VCF store. Being a table, the associated permissions apply to the sample-participant IDs mapping, i.e. a user could see the VCF files without seing the sample-participant mapping table.

View VCF files statistics Permission

View VCF files and statistics. No possibility to export the VCF files.

View VCF files data and statistics Permission

Export VCF files and view statistics.

Administrate VCF Store

Import/Export VCF files, view statistics and set samples-participants mapping table.

Operations

Import

Import a VCF/BCF file from the Opal file system into the project's VCF store. The VCF/BCF file can be optionally compressed (by *bgzip*). If not compressed, the process of importation will compress it. In addition to that the file will be indexed, some general summary statistics and the sample list will be extracted.

> The VCF/BCF files must be sorted. The following command could do the trick:
>
> ```
> cat plate.vcf | grep ^# ; cat plate.vcf | grep -v ^# | sort -k1,1d -k2,2n > plate.sort.vcf
> ```

Export

This operation will write the selected VCF files into the Opal file system. The proposed export options will depend on wether a sample-participant mapping table has been defined or not.

If a sample-participant mapping table is defined, the export options are:

- export only the samples which are associated to the set of participants being entities of Participant table,
- export the case controls.

Add Mapping

The sample-participant mapping table is necessarily about `Sample` (expected entity type) and can be in the current project or another one. The variables that will provide the participant ID and the sample role (*sample* or *control*) are to be identified.

Project Files

Overview

Summary

Each project has its own folder in Opal's file system, located at path `/projects/<project name>`.

The information available and the operations that can be performed are the same as what is accessible from the Files Administration page.

Permissions

In terms of permissions, only users having write permissions over the project itself or a table will be granted read and write permissions over the project's home folder.

Project Reports

Overview

Summary

Reports can be written in R Markdown format and associated to a project. This functionality requires the R server to be properly configured, i.e. with rmarkdown package installed within some system libraries (pandoc, tex). See R Server Installation Guide.

Operations

Add Report

Design a report in R Markdown (see R Markdown Tutorial), upload it in Opal file system, and select it as the report template.

The output format should match the one specified in the R Markdown document, so that Opal can find the generated report file.

Some R options can be specified: these key value pairs will be pushed in the R server session as R options to be available during R Markdown rendering task.

Download Report Design

The report template can be downloaded individually.

Execute Report

Manually execute the report.

Delete Report

Delete the current report design and reports that were produced.

Project Tasks

Overview

Summary

Project Permissions

Overview

Introduction

Project Administration

Overview

Summary

The project administration section is only accessible to project or system administrators. Project specific description and operations are available.

Properties

This section allows to change the project's defaults properties. The following properties can be modified:

- The title is the human readable name of the project.
- The database is the datasource associated to the project. It is not allowed to change the database for a project that contains data.
- The VCF Store is the name of the service that is used to handle the VCF/BCF files of the project. This property might not be available as this service is a Opal plugin.
- The description of the project consists of a brief description of the project that will be shown on the projects list page.
- A list of tags that would eventually be useful to search or filter projects from the projects list page.

Encryption Keys

To protect participants privacy imported files can be encrypted. The importation process of encrypted material is the following:

- find the encryption private key associated to the certificate that was used to encrypt the file by looking in the project's keystore,
- decrypt in memory the file.

Import Key Pair

Add an encryption key for the project from an existing private-public key pair in PEM format.

Create Key Pair

Creates an encryption key to the project.

Permissions

Specify the access rights to a particular project and its content. See also Project Permissions.

Danger Zone

Archive Project

This section allows to archive a project without destroying all associated data and files. Once a project is archived, recreating a project with the same name using the same database will reinstate the associated tables.

Remove Project

This section allows to completely remove a project by destroying all associated data and files. Once a project is deleted, it cannot be recovered.

Searching

Contents of this Guide

Summary

This guide provides a description of the web interface for searching Opal content. See more specific search sections.

- Search Variables
- Search Entity
- Search Entities

Search Variables

Overview

Summary

Variables can be searched using controlled vocabularies or full-text search.

Controlled Vocabularies

The controlled vocabularies are the ones defined by the taxonomies and the variable properties. Once a vocabulary term has been selected, it will be used for filtering the variables. Given such a criterion different filters can be selected:

Operation	Description
`Any`	The variable field can have any non null value.
`None`	The variable field must be missing.
`In/Like`	The variable field must be in: • In: at least one of the selected predefined values • Like: match a query string (with wildcard support)
`Not in/Not like`	The negation of `In/Like`.

Taxonomies

See Taxonomies Administration documentation. As the number of vocabulary terms can be very large, the interface allows to search for these terms (name, label, description) by providing keywords. These keywords can be negated, for instance:

```
alcohol -constructs
```

will look up taxonomy terms containing the word `alcohol` AND NOT containing the word `constructs` in its name/label/description (or in the name/label/description of the associated vocabulary).

Properties

The variable properties that can be used are:

Property	Description
`project`	The project the variable belongs to
`table`	The table the variable belongs to
`name`	The name of the variable.
`entityType`	The type of the entity: Participant, Sample etc.
`valueType`	The type of the variable values: text, integer, decimal etc.
`nature`	Nature of the variable: categorical, numerical, logical etc.
`repeatable`	A variable is repeatable when it can have several values for one entity.
`occurrenceGroup`	When a repeatable variable is in the same group of occurrence as other repeatable variables
`referencedEntityType`	When the values of the variable is an identifier, this property specifies what is the type of the referred entity.
`mimeType`	The mime type of the data.
`unit`	The measure unit.
`script`	The variable attribute that holds the derivation script.

The property lookup will be done on the property name or on its possible values. For instance:

```
nature
```

will propose to choose among all the variable nature values (categorical, numerical etc.). Whereas typing:

```
categorical
```

will propose the categorical nature only.

Full-text Search

The full-text search applies to:

- the variable name,
- the variable label(s) in any language.

Wildcard can be used.

Advanced Search

The advanced search option allows to define your own query. See Elasticsearch Query Syntax for detailed explanation. We recommend to use the controlled vocabulary first to get the corresponding field names that are not necessarily obvious and then combine criteria at will by using AND, OR and NOT conjunction words.

Results

The resulting variables are presented as a list. To make this list useful it is possible to select some variables and add them to the global Cart. Once in the cart the variables, that could be the result of several search, can be used to search for entities or make a view from them, etc.

Search Entity

Overview

Summary

Each entity data can be displayed by providing the type and the identifier of this entity.

Results

If the entity exists in the given type, the values of this entity will be displayed one table at a time.

Filter variables

A quick filter allows to show only variables of interest. For instance typing:

```
alc -comment
```

will show only the variable with name containing `alc` AND NOT containing `comment`.

Show empty values

The variables for which there is no value can be hidden.

Search Entities

Overview

Summary

Entities can be searched by defining variable criteria. The result of this search gives the count of entities for each of the criterion and the count of entities satisfying all the criteria. The variables can be from different tables, meaning that the resulting count is the intersection of each entity sub-query.

Prerequisite

Only tables which values have been indexed can be searched.

Variable Criteria

A variable criterion, is a variable which will be used to discriminate the entities satisfying some constraints on its values. Several variable criteria are combined with AND conjunction: the results must satisfy each criterion.

Add Criterion

Lookup and Add

The variable of interest can be found by typing keywords such as:

```
alco wine -weekend
```

will propose variables containing `alco` AND `wine` in their name/label AND NOT containing `weekend`.

Add from Cart

The cart can be populated with variables that where searched (see Search Variables) or that were added when exploring the table (selection from the variable list of the table page, or individual selection from the variable page).

Use Criterion

Each variable criterion can be used to filter the entities:

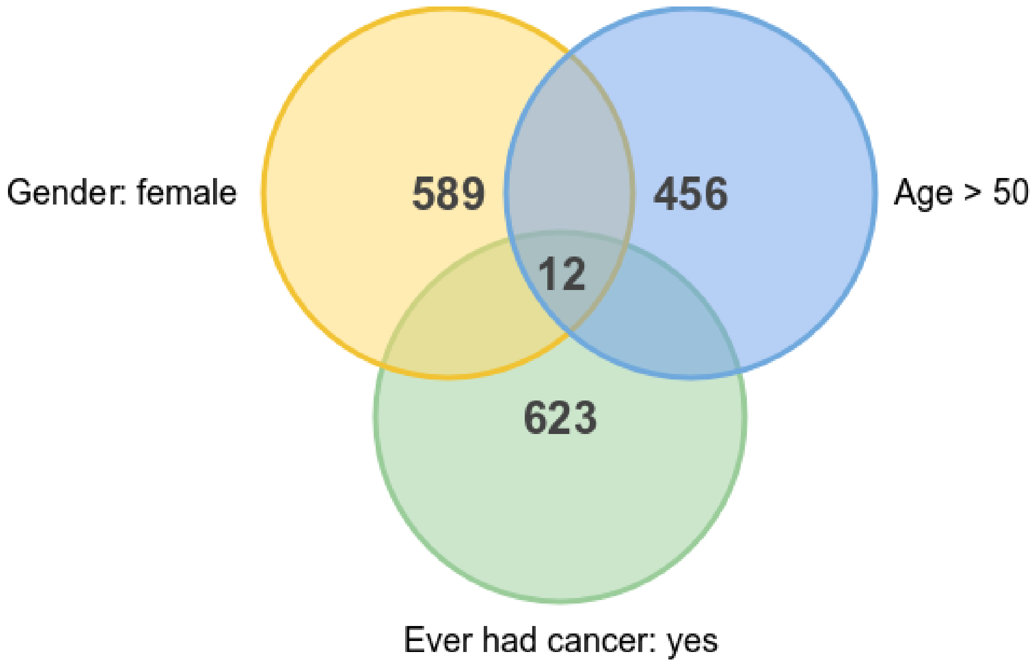
Operation	Description
`All`	The values of the variable can be any value (empty or not empty).
`Empty`	The values of the variable are empty (null value or value sequence with all null values).
`Not empty`	The values of the variable are not empty (not null value or value sequence with at least one not null value).
`In`	The value of the variable has some specified value(s).
`Not in`	The value of the variable has not some specified value(s).

Identifier Criterion

It is also possible to filter the entities by their identifier. Exact match or wildcard can be used to specify this filter.

Results

The count of entities matching each of the variable criteria will be displayed. The count of entities satisfying **all** the criteria will be provided as well. This can be illustrated by the venn diagram:



A view can be built with a entities filter script that reproduces the search criteria.

Administrating Opal

Contents of this Guide

Summary

This guide provides a description of the web interface for administrating Opal. See more specific administration sections.

- Data Access
- Data Management
- Data Analysis
- System

Databases

Opal can connect to different types of databases (SQL, No-SQL), with different data schema and for different purposes (import/export, storage). For more details see the Databases Administration documentation.

R

In order to configure Opal for R analysis you can refer to:

- Opal R and DataSHIELD User Guide,
- R Administration.

DataSHIELD

In order to configure Opal for DataSHIELD analysis you can refer to:

- Opal R and DataSHIELD User Guide,

- DataSHIELD Administration.

Search

Opal embeds an data indexing engine for search purpose: Elastic Search. See details on how to administrate Elastic Search in Search Administration documentation.

Data Access

- Users and Groups Administration
- Profiles Administration

Users and Groups Administration

Overview

- Summary
- Users
- Groups
- Operations
  - Add User with Password
  - Add User with Certificate
  - Edit User
  - Remove User
  - Disable User
  - Remove Group

Summary

Opal has its own user directory, accessible from the users and groups administration page.

Users

There two types of user in Opal, depending on the way they are authenticated:

- by password: this is the most standard way, allowing a physical user to connect to the web interface,
- by certificate: this is convenient when an application needs to connect to Opal server. This only works when connecting through `https`: Opal identifies the user by comparing the received certificate with the registered one and encrypts in return the communication with this certificate. Only a remote client having the corresponding private key (not known by Opal) will be able to decrypt the response. For more information see Client-authenticated TLS handshake documentation. In order to have this handshake to work, the SSL headers must not be altered by proxies, firewall or load balancers that could be between the client application and the Opal server.

Groups

A group is just a group of users, i.e. no group can be created without users.

Operations

Add User with Password

At least 6 characters are required for the user password. After a user has been added, a personal folder is added in the `home` folder of the Opal's file system.

Add User with Certificate

The user X.509 certificate (or public key) must be provided in PEM format.

Edit User

Change password or X.509 certificate or group membership.

Remove User

Remove a user and all the permissions that could have been granted to this user. The home folder of this user (located at the path `/home/<user name>` in Opal's file system is untouched).

Disable User

User and its associated permissions are still available but user cannot login anymore.

Remove Group

Removing a group consist of excluding associated users from this group.

Profiles Administration

Overview

Summary

A user profile hold some information about the user:

- directory origin (allows to prevent users with same name but from different directories to login),
- user timestamps (first and last seen dates)
- user bookmarks

Operation

Remove

Removing a profile can be useful for resolving a user directory conflict (something that is very unlikely to happen).

Data Management

Files Administration

Overview

Summary

This guide provides a description of the web interface for browsing, uploading and downloading files on the server.

File System

View

The file browser consists of two parts: folder shortcuts on the left and directory/file listing on the right. The default location is the current user's home folder.

Operations

An administrator can perform the following operations on all folders whereas a user with limited privileges can only perform them in the user's home directory.

Add Folder

Adds a new sub-folder in the current location.

Upload

Uploads a file in the current location.

Download

Downloads the selected files and folders to the user's computer. If there are more than one file or directory selected, a ZIP containing all of them is downloaded to the user's computer.

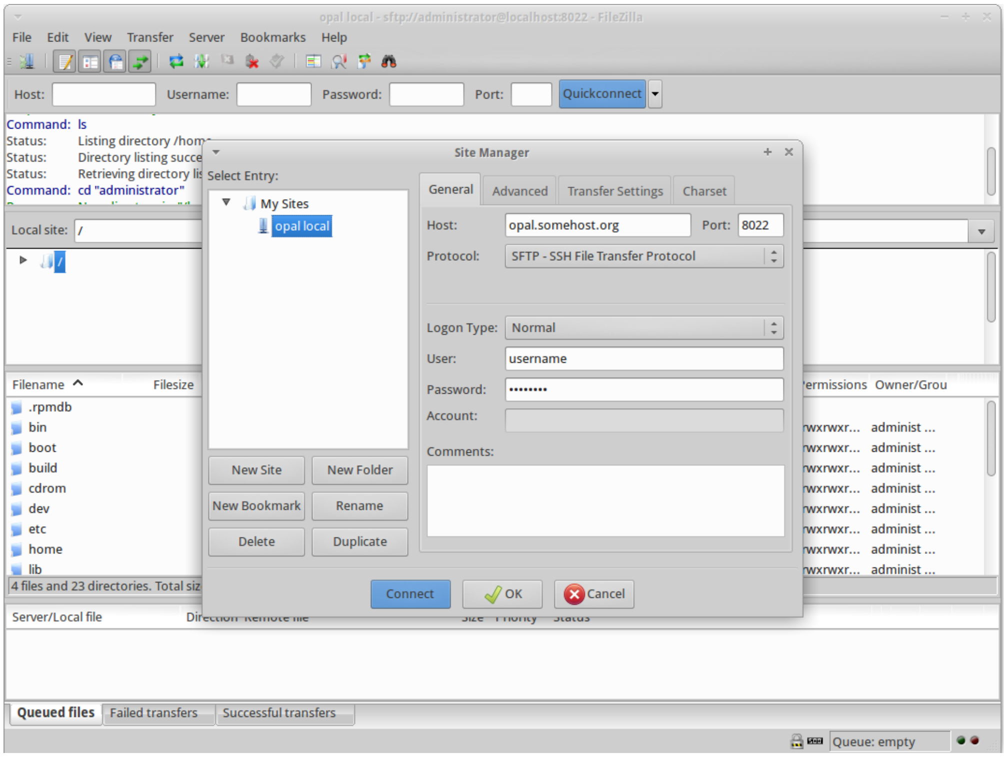
Delete

Deletes the selected files and folders from the current location.

SFTP

Users can securely access Opal's file system using SFTP with any third-party tools such as FileZilla or the Firefox add-on FireFTP. Information required to configure FileZilla:

- Host: IP or the host name where the Opal server is running
- Protocol: SFTP
- Port: 8022
- Logon Type: Normal (requires a valid username and password)



opal local - sftp://administrator@localhost:8022 - FileZilla

```
Command:  ls
Status:   Listing directory /hom
Status:   Directory listing succe
Status:   Retrieving directory lis
Command:  cd "administrator"
```

Site Manager

Host: opal.somehost.org Port: 8022

Protocol: SFTP - SSH File Transfer Protocol

Logon Type: Normal

User: username

Password: ••••••••

Account:

Comments:

4 files and 23 directories. Total siz

Queue: empty

File Selector

File Selector enables the user to explore the Opal file system.

Operations

Add Folder

Adds a new sub-folder in the current location.

Upload

Uploads a file in the current location.

Tasks Administration

Overview

Summary

This guide provides a description of the web interface for listing and managing tasks that were launched on the server.

> **Documentation is in progress**

Tasks

Operations

Refresh

Clear

Task Details

Operations

Log

Cancel

Reports Administration

Overview

- Edit

Summary

This guide provides a description of the web interface for scheduling reports execution. The reports are to be designed in R Markdown: see the tutorial for designing such a report in Opal Reporting User Guide.

Operations

Add Report Template

The Report Template Name must be unique. The R Markdown Report Design File is required.

Options, Scheduling and Notification Emails are optional.

Report Template Details

Report Template Name

This name of the report template is unique.

R Markdown Report Design File

This file was created and maintained using a editor such as RStudio. For more information see Opal Reporting User Guide. The file must be accessible in the Opal file system. Moving the file location after the report template is created will prevent the report to be successfully generated.

Options

Some R Markdown reports are designed with R options. These options will be pushed to the R session at the time of the report execution.

Some known opal.login and opal.report related R options are:

R option	Description
`opal.username`	Name of the user that will connect to the Opal server
`opal.password`	Password of the user
`opal.url`	Opal server address
`opal.report.style`	One of the report style defined by knitrBootstrap R package: Default, Amelia, Cerulean, Cosmo, Cyborg, Journal, Flatly, Readable, Simplex, Slate, Spacelab, United

Any other user defined R options are accepted.

Scheduling

A schedule expression is a string comprised of 6 or 7 fields separated by white space. Fields can contain any of the allowed values, along with various combinations of the allowed special characters for that field.

Full documentation is available as a Quartz Scheduler Tutorial.

Field Name	Mandatory	Allowed Values	Allowed Special Characters
Seconds	YES	0-59	, - * /
Minutes	YES	0-59	, - * /
Hours	YES	0-23	, - * /
Day of month	YES	1-31	, - * ? / L W
Month	YES	1-12 or JAN-DEC	, - * /
Day of week	YES	1-7 or SUN-SAT	, - * ? / L #

Year	NO	empty, 1970-2099	, - * /

Common schedule expressions examples:

Expression	Meaning
0 15 10 ? * *	Fire at 10:15am every day
0 15 10 ? * MON-FRI	Fire at 10:15am every Monday, Tuesday, Wednesday, Thursday and Friday
0 15 10 15 * ?	Fire at 10:15am on the 15th day of every month
0 10,44 14 ? 3 WED	Fire at 2:10pm and at 2:44pm every Wednesday in the month of March.
0 15 10 ? * 6#3	Fire at 10:15am on the third Friday of every month

Notification Emails

A list of emails to whom a message will be sent each time a new report is generated. The message will contain a public link to the report for downloading it. It is not necessary to have an account in Opal to be able to download the generated reports.

Reports

The list of reports associated to the report template. Also directly available from the Opal file system in `/reports/<report_template>` folder.

Operations

Run

Execute the report manually. The report job execution can be followed from the Tasks Administration page. Once the report is available it can be accessed through the corresponding report template report list or directly in the Opal file system.

Remove

Remove the report template (associated reports are not removed and will be still available from the Opal file system in `/reports/<report_template>` folder).

Edit

Modify the report template details. Same dialog as Add Report Template except that the template name cannot be modified.

Identifiers Mapping Administration

Overview

Summary

This guide provides a description of the web interface for managing units, i.e. participant identifiers and encryption keys.

Unit, Encryption and Identifier

In data files to be imported, participants have an identifier that is valid in the scope of the data provider. This data provider is formalized in Opal as a Functional Unit which is to be created before performing the data importation task. The unit will also hold the Encryption Keys to be used for decrypting the data files coming from this data provider.

When exporting data, the same operations are performed in a reversed order, i.e. instead of exporting participants data with their Opal internal identifier, selected unit identifiers are used. To achieve that, the exportation target is formalized as a Functional Unit. After having defined the participants identifiers for the targeted unit, Opal is able to export data without exposing internal identifiers.

When handled properly, the unit identifiers can allow exchange of participants data, back and forth, between Opal and another unit:

- without exposing Opal internal participant identifiers,
- by making Opal the controller of the data exchanged between two units.

The unit web application interface can be used to manage the units declared in Opal:

- Encryption Keys,
- identifiers mapping and export,
- data importation/exportation related operations.

See Identifiers Mappings for a general description of what is a unit.

Operations

Add Unit

The Unit Name must be unique in the system.

The #Identifiers Variables Filter is optional.

Export All Identifiers

Exports all the identifiers registered in Opal as a CSV file. Each column name is a Unit Name. The first column is `OpalInstance`, Opal being a unit by it-self. Each row is a participant, with its identifiers for each unit (when defined).

Map Identifiers

The process of mapping identifiers requires to provide a CSV file with the following content:

- exactly two columns,
- each column name is a Unit Name,
- each row is a participant and data are identifiers.

unit1	unit2
1000	a
1001	b
1002	c

After having provided this file, Opal tries to parse it and to identify the units by the column names. If successful, user is asked to select which unit is to be used for retrieving the participants (the unit that "drives" the identifiers mapping). Every identifiers of the driving-unit must have been registered before.

Then for each driving-unit identifier, the participant is fetched and is affected a new identifier for the targeted unit. If an error occurs, no identifier is added.

This operation cannot be used to update an identifier mapping. Once an identifier is assigned to a participant for a unit, it cannot be modified.

After identifiers mapping is done, the Current Count of Identifiers of the targeted unit is increased with the number of added identifiers.

Synchronize Identifiers

Some datasources could have entities with an identifier that is unknown in the identifiers database (usually after an additional database has been connected to Opal). The operation of synchronization will import them in the identifiers database. The identifiers cannot be related to a unit: they are all considered as Opal identifiers.

Before the operation is performed, the user will be shown the count of identifiers that are not in the identifiers database for each table.

Unit Details

Unit Name

The name of the unit identifies the unit uniquely. The name is used when importing/exporting data and when performing identifiers mapping.

`OpalInstance` is the reserved unit name for Opal (which is is a unit by it-self).

Identifiers Variables Filter

When importing data, in addition to participant identifiers some data can allow the identification of participants: for instance participant name, address, phone number etc. This kind of data must not be imported as a regular data. It must be filtered out and imported with the participant identifiers in the identifiers database (see general information about Participant Identifier Separation).

To perform this, a script can be provided written with the Magma Javascript API. This script will select the variables that are participant identifiers. The importation process will behave accordingly.

Current Count of Identifiers

Reports the number of participants having an identifier in the unit.

Encryption Keys

A set of encryption keys can be associated to a unit. These encryption keys are used to:

- publish certificates used for encrypting data,
- decrypt in return the data at importation time.

Each encryption keys is a pair of a private key and a certificate (available for download).

Operations

Add Encryption Keys

A new key pair can be added:

- a new private key is created, and then a new certificate is to be created too.
- or the private key is imported (as a PEM format text), and then:
    - a new certificate can be created associated to the provided private key,
    - or the corresponding certificate is imported (as a PEM format text).

Generate Identifiers

A new identifier is generated and assigned to each participant not having an identifier in the considered unit. For a better control of the unit identifiers, use the Map Identifiers operation instead.

The pattern for identifier generation supports some options:

- identifier size,
- identifier prefix: each generated identifiers will start with this prefix,
- allow leading zeros: if random number is shorter than the identifier size, leading 0s will be added.

Add Identifiers From Data File

This is similar to importing data except that only participant identifiers are handled. Given an identifier in the considered unit, if no participant can retrieved, a new participant is created. The Identifiers Variables Filter will be applied too. If the data file is encrypted, the Encryption Keys should have been defined first.

Export Identifiers

Export all the unit identifiers in a text file, one row per identifier.

Export Identifiers Mapping

Export all the unit identifiers mapped with `OpalInstance` unit, one row per participant.

Remove

Removes the unit description and the Encryption Keys. Does not remove the associated identifiers from the identifiers database. If a unit is created afterwards with the same name, the identifiers will be visible again.

Data Analysis

- R Administration
- DataSHIELD Administration
- Search Administration

R Administration

Overview

- Introduction
- R Server
- R Sessions
- Permissions
  - Use

Introduction

Opal allows to perform data analysis on the R server. User or group #Permissions for this service can be granted. See the Opal R User Guide for detailed usage of the R service.

R Server

Allows to start and stop the R server. Stopping the R server will destroy all R sessions and workspaces currently active. After installing R packages, it could be required to restart the R server in order to have the changes loaded in the main R environment.

R Sessions

The list of active R sessions is available. An administrator can destroy and R session resulting the corresponding remote R client to fail subsequent commands.

Permissions

Specify the access rights to the R services.

Use

This permission is to be used for allowing R clients (with Opal-R package) to access to R services.

DataSHIELD Administration

Overview

- Summary
- Packages
  - DataSHIELD Repositories
  - Installing Packages
  - Publish Package Methods
- Options
- Methods
  - Aggregation Methods
    - Aggregation R Function
    - Aggregation R Script
  - Assign Methods

Summary

Opal allows DataSHIELD to be performed on the set of Methods that have been published. DataSHIELD-R packages can be installed in R server. User or group Permissions for this service can be granted.

Packages

Opal is able to install DataSHIELD R packages in the connected R server.

> **R server**
>
> - R server must run on behalf of a user having access to a home directory in order to allow packages to be installed,
> - R version must be >= 2.15 to allow DataSHIELD packages installation from source repository.

DataSHIELD Repositories

The DataSHIELD packages are available from:

- source repository,
- package CRAN repository,

When installing a package, if a GIT reference is provided (can be a branch, tag, commit number), the corresponding source code will be downloaded from the source repository and installed as a package in R server.

Installing Packages

If R server is correctly set up, installation of DataSHIELD R packages should be straight forward:

- select "Add Package"
- option is to install all R DataSHIELD packages (default), or the name of the package to be installed can be specified: `dsmodelling`, `dsbase or dsteststats`

Once successfully installed, the package methods are automatically published.

> If the installed packages cannot be seen in the packages list, it might be due to:
>
> - either the R server runs on behalf of a user that does not have the permission to install R packages (requires a home directory)
> - or the server does not have access to CRAN repositories: `http://cran.obiba.org` and `http://cran.rstudio.com`
> - or the packages were correctly installed but the R server process does not have included the newly created R library directory in its libraries look-up paths. Restarting R server will fix this issue.

Publish Package Methods

Each DataSHIELD package comes with a set of Aggregation R Function and Assign R Function. Publishing those methods will make them executable from a DataSHIELD client.

Options

Some R options can be specified. Each time a R session is created for DataSHIELD, these options will be applied and made available to the executed R commands. Some default DataSHIELD options are extracted from the DataSHIELD packages at installation time. The value of these options can be changed afterwards.

Methods

Aggregation Methods

The aggregation methods are used by DataShield in order to compile individual data. The same aggregation methods must be defined in each DataShield server that will be involved in a computation process. Each aggregation method is identified by a name that will be used from the

R-DataShield client.

Aggregation R Function

Some aggregation methods are already defined in R as functions. For security reasons, R functions should be fully named, i.e. including package name space. This way the aggregating function to be used can be unambiguously identified. For example use "base::summary" in place of "summary".

Aggregation R Script

Any R script that does not return any individual data.

Assign Methods

The assign methods are used by DataShield in order to transform individual data on server side. The same assign methods must be defined in each DataShield server that will be involved in a computation process. Each assign method is identified by a name that will be used from the R-DataShield client.

Assign R Function

Some assign methods are already defined in R as functions. For security reasons, R functions should be fully named, i.e. including package name space. This way the assigning function to be used can be unambiguously identified. For example use "base::data.frame" in place of "data.frame".

Assign R Script

Any R script that transforms individual data.

Permissions

Specify the access rights to the DataShield services.

Use

This permission is to be used for allowing R clients (with Opal-R package) to access to DataShield services.

Administrate

Allow the administration of the DataShield configuration, i.e. the aggregation methods management.

Search Administration

Overview

Summary

Opal uses Elasticsearch (ES) to index dictionaries and data. ES can be start/stopped and configured.

Search Service

Start/Stop and Configure Elasticsearch.

By default Elasticsearch is started with the settings that are defined in the source file default-settings.yml.

These can be overridden in the settings section of the Elasticsearch configuration form.

Elasticsearch Nodes

Elasticsearch can be connected to other nodes for load balancing. The nodes discovery strategies can be configured. By default `multicast` disc overy is disabled; `unicast` discovery is recommended for production.

By default Opal embeds an Elasticsearch server that is self sufficient for small datasets. In order to reduce the memory footprint of Opal and to scale larger needs, a separate Elasticsearch server can be installed (on the same or on a different machine) and the embedded Elasticsearch can be used as a proxy to this external server.

The following example explains this configuration.

External Elasticsearch server configuration

1. Install Elasticsearch server
2. Configure it (edit the file `/etc/elasticsearch/elasticsearch.yml` on linux)
    - specify the same cluster name as the one of Opal's Elasticsearch
    - disable multicast discovery
    - specify Opal's token analyzers and filters
3. Restart Elasticsearch server

```
cluster.name: opal
discovery.zen.ping.multicast.enabled: false
discovery.zen.ping.unicast.hosts: ["opalhost"]
index:
    max_result_window: 1000000
    analysis:
        analyzer:
            opal_index_analyzer:
               type: custom
               char_filter: [html_strip]
               tokenizer: standard
               filter:
[standard,lowercase,opal_asciifolding_filter,opal_nGram_filter]
            opal_search_analyzer:
               type: custom
               char_filter: [html_strip]
               tokenizer: standard
               filter: [standard,lowercase,opal_asciifolding_filter]
        filter:
            opal_asciifolding_filter:
                type: asciifolding
                preserve_original: true
            opal_nGram_filter:
                type: nGram
                min_gram: 2
                max_gram: 20
```

Embedded Elasticsearch configuration

Example of a `unicast` configuration:

1. Edit Elasticsearch configuration
    - specify the same cluster name as the one of the external Elasticsearch
    - specify that the embedded Elasticsearch does not hold data
    - specify where to find the external Elasticsearch server (see node discovery documentation)
2. Restart Elasticsearch service

```
node.data: false
discovery.zen.ping.unicast.hosts: ["eshost"]
```

Elasticsearch Head

Elasticsearch Head is a standalone page that allows to browse and query your Elasticsearch node. In order to use it with Opal, the following setting is required:

```
http.enabled: true
```

Variables Index

Each variable description is automatically indexed (~1min of delay) after a table is updated (import or edition). When this index is removed, a new index is automatically recreated with all variables. Removing this index is necessary after the system languages have been modified.

Values Index

List of all tables available in Opal datasources with the status of their respective values index.

Each table index can be scheduled individually:

- manually,
- every 5 minutes,
- every 15 minutes,
- every 30 minutes,
- every specific minute in hours,
- every specific time in days,
- every specific day time in weeks.

System

- General Settings Administration
- Taxonomies Administration
- Databases Administration
- Plugins Administration
- Java Virtual Machine Monitoring

General Settings Administration

Overview

- Summary
- Properties
- Encryption Keys

Summary

This section is about system level configuration. Accessible only by an administrator.

Properties

Property	Description
Name	Name of the Opal server that will be displayed in the web interface. Helps to distinguish several Opal instances.
Default Character Set	When reading/writing files, if a character set is not specified, Opal defaults to **ISO-8859-1**. This is used for example when reading/writing CSV files.
Public URL	Public base URL of the server (not the web-interface one) that will be used when sending notification emails on report generation.

Language	Default languages to consider when editing a data dictionary.

Encryption Keys

HTTPS connection requires to have a private key and a public key (certificate) defined. A self-signed key-pair is available by default. You can provide your own. Opal server needs to be restarted after the encryption keys have been updated.

Taxonomies Administration

Overview

Summary

Taxonomies are used to perform variables classification. Taxonomy items (vocabulary and term) have a title and a description (multi language support).

Taxonomy

A taxonomy is a set of controlled vocabularies. It provides also authoring information (author, license). Recommended license is one of the Creative Commons licenses.

Vocabulary

A vocabulary is controlled in the way that it provides a set of terms. These terms are used to *annotate* the variables: a variable annotation is a variable attribute which namespace is a taxonomy, name is a vocabulary and value is one of the terms defined by the vocabulary.

When a vocabulary has no term, any text is accepted as a variable annotation for this vocabulary. Opal supports text formatted in Markdown.

Operations

Add Taxonomy

Add a taxonomy from scratch.

Import Maelstrom Research Taxonomies

Maelstrom Research provides a complete set of taxonomies to classify variables (classification based on the experience of more than 500K variables) and to describe the dataset harmonization process. Importing these taxonomies requires a download key that can be requested at Maelstrom Research (link to request form is provided).

Import Taxonomy from Github

Import one or more taxonomies from a Github repository. Taxonomy YAML files are expected to be found in this repository. A taxonomy YAML file is the one that can be downloaded from a taxonomy page.

Databases Administration

Overview

Summary

The databases administration page allows to manage the server databases. A fully operational Opal server requires to have at least two different databases registered for:

- identifiers mapping storage (one and only one required, see Identifiers Mappings section for more details)
- data storage (at least one is required)

Additional databases can be declared for other usages: data import, data export.

Opal currently supports two different type of database engines:

- SQL database (MySQL, PostgreSQL) for storage, import, export,
- Document database (MongoDB) for storage only.

The following table summarizes the different database usages depending on the database engine and the schema used to store the data.

Database Engine	Data Schema	Storage	Import	Export
MySQL	Opal SQL	✓	✗	✗
MySQL, PostgreSQL	Tabular SQL	✓	✓	✓
MySQL, PostgreSQL	Limesurvey	✗	✓	✗
MongoDB	Opal Documents	✓	✗	✗

Database Engines

SQL Databases

Currently the supported SQL database engines are: MySQL and PostgreSQL. Make sure the corresponding database users are granted all privileges on their respective database instances (CREATE TABLE, ALTER, and so on).

MySQL

At the time of writing this document, at least **MySQL 5.5.x** is recommended.

MySQL Server Configuration

Edit the `my.cnf` file (often named `my.ini` on Windows operating systems) in your MySQL server. Locate the `[mysqld]` section in the file, and add or modify the following parameters:

- specify the default character set to be UTF-8:

```
[mysqld]
character-set-server=utf8
collation-server=utf8_bin
```

- set the default storage engine to InnoDB:

```
[mysqld]
default-storage-engine=INNODB
```

- if you plan to store binary data into Opal, configure the packet size that wil be transmitted to or from MySQL. See Packet Too Large documentation.

```
[mysqld]
max_allowed_packet=1G
```

- we also recommend to use Per-Table Tablespaces. See InnoDB File-Per-Table Tablespaces documentation.

```
[mysqld]
innodb_file_per_table
```

MySQL Database Creation

When creating the MySQL database that Opal should connect to, make sure the character set is specified as **UTF-8** with binary UTF-8 collation (for case-sensitive collation).

```
CREATE DATABASE opal CHARACTER SET utf8 COLLATE utf8_bin;
```

The default MySQL storage engine must also be InnoDB.

Sample script for MySQL database creation:

create_opal_database.sql

```
# Create Opal database and user.
#
# Command: mysql -u root -p < create_opal_database.sql
#

CREATE DATABASE opal_data CHARACTER SET utf8 COLLATE utf8_bin;

CREATE USER 'opal' IDENTIFIED BY '<opal-user-password>';
GRANT ALL ON opal_data.* TO 'opal'@'localhost' IDENTIFIED BY
'<opal-user-password>';
FLUSH PRIVILEGES;
```

PostgreSQL

PostgreSQL is currently supported for all usages associated with the Tabular SQL schema (import/export and storage). Limitations associated with this type of schema applies.

Document Databases

Currently the only No-SQL engine that is supported is the document oriented database MongoDB.

MongoDB

MongoDB does not require the database to exist before you access it. So you could just install MongoDB and configure your database in Opal.

It is however recommended that you restrict access to your MongoDB database, to achieve this you need to:

- create a user with the proper roles on the target databases
- run the MongoDB service with Client Access Control enabled

> Once the MongoDB service runs with Client Access Control enabled, all database connections must be authenticated.

- specify the authentication source database in the connection URL:

Connection URL Examples

```
mongodb://localhost:27017/opal_ids?authSource=admin
mongodb://localhost:27017/opal_data?authSource=admin
```

The example below creates the *opaladmin* user for *opal_ids* and *opal_data* databases:

MongoDB Console

```
use admin
db.createUser(
  {
    user: "opaladmin",
    pwd: "opaladmin",
    roles: [
      {
        "role" : "readWrite",
        "db" : "opal_ids"
      },
      {
        "role" : "dbAdmin",
        "db" : "opal_ids"
      },
      {
        "role" : "readWrite",
        "db" : "opal_data"
      },
      {
        "role" : "dbAdmin",
        "db" : "opal_data"
      },
      {
          "role": "clusterMonitor",
          "db": "admin"
      },
      {
          "role": "readAnyDatabase",
          "db": "admin"
      }
    ]
  }
)
```

Opal requires either *clusterMonitor* or *readAnyDatabase* role on the *admin* database for validation operations. The first role is useful for a cluster setup and the latter if your MongoDB is on a single server.

Data Schemas

Depending on the database engine and usage, an administrator will be asked to specify how the data will be organized in the database. See Variables and Data documentation for a description of the Opal's data model. This data model can be persisted in different data schemas depending on the usage.

Opal SQL

The purpose of this SQL data schema is to be able to accommodate any number of variables from the Opal table abstraction point of view. A SQL-table will have a limit in terms of number of columns that can be added (this limit depends on the database engine). The *Opal SQL* schema follows the Entity-attribute-value model (EAV), which allows to describe Opal tables with thousands of variables. However the price of the EAV schema is that querying data requires a lot of SQL join requests. Opal tries its best by caching SQL query results but there is still a performance price for this flexibility.

You may choose this data schema when:

- the number of expected variables is large (more than several hundreds),
- flexibility is preferred to performance.

Tabular SQL

The *Tabular SQL* schema propose a more standard representation of the data: there is one SQL table per Opal table (and therefore one column per variable). Querying such schema is very straightforward but data persistence has some limits:

- the number of columns in a SQL table and/or the size of each row are limited (and therefore the number of variables in a Opal table). This number depends on the database engine. In the case of MySQL there is a hard limit of 4096 columns per table but the effective limit depends on the size of the rows that are being persisted. For more information see Limits on Table Column Count and Row Size in MySQL documentation or the About PostgreSQL documentation.
- the name conflicts between variables (resp. tables) are more likely to occur as characters used for naming objects and length of the names are limited: see Schema Object Names and Identifier Case Sensitivity in MySQL documentation or Identifiers and Key Words in PostgreSQL documentation.
- the generated SQL type may not be optimal for some data. For instance the text type does not have data length constraint: this affects the row size although some data could be short text. Also binary values are stored in a column with BLOB (or bytea) type which data size can be limited.

On the other hand this data schema still worth to be chosen when:

- the number of variables is limited (less than several hundreds, modulo the data size of each row),
- queries involving vector need to be fast (data summary of a variable, assignment to a R dataframe),
- import of an existing SQL table,
- export to a SQL table.

Opal offers to specify some settings for this schema:

Setting	Description	Remark
Entity Identifier Column	Name of the column containing the identifier of the entity in the SQL-table. This column will not be considered as a variable. This identifier column is a primary key, i.e. there must be only one row with a given identifier (same rule applies to a CSV file). Only the SQL-tables having this column can be mapped to a Opal table.	Required, value is *opal_id* when usage is storage.
Creation Timestamp Column	Name of the column holding the timestamp of the creation of a row in the SQL-table. This is a purely informative information that makes sense only when data are subsequently updated.	Optional, value is *opal_created* when usage is storage.
Update Timestamp Column	Name of the column holding the timestamp of the last modification date of a row in the SQL-table. This information can be useful when performing an incremental import (only new or updated rows are imported).	Optional, recommended for import/export, value is *opal_updated* when usage is storage.
With variables description tables	In addition to the SQL tables of data, the data dictionary can be persisted in other SQL tables: *value_tables*, *variables*, *variable_attributes*, *categories* and *category_attributes*. This allow to have fully described data (otherwise the data dictionary is limited to the column names and SQL types).	Optional, recommended for import/export, selected when usage is storage.
Default Entity Type	When there is no variables description tables, this setting specifies the entity type of the tables that are discovered.	Required.

The mapping beween the SQL types and the Opal value types is the following:

SQL Type	Value Type

BIGINT, INTEGER, SMALLINT, TINYIN	integer
DECIMAL, DOUBLE, FLOAT, NUMERIC, REAL	decimal
DATE	date
TIMESTAMP	datetime
BIT, BOOLEAN	boolean
BLOB, LONGVARBINARY, VARBINARY, BINARY	binary
anything else	text

Limesurvey

Opal is able to read directly the SQL data schema of a Limesurvey server. Opal will detect the completed interviews and will import the new and updated ones. The variables are also extracted from the Limesurvey questionnaire.

Operations

Register

Registering a database requires to specify:

- the database engine,
- a unique name for identification when creating a project or importing/exporting,
- the connection details: jdbc url and credentials (user name, password),
- the usage (applies to SQL database engine only),
- the data schema (applies to SQL database engine only, choice is limited by selected usage),
- optional properties (key, value pairs).

Depending on the database engine, the declared usage and the data schema some options may be available or not.

Several databases can be registered for storage usage. All databases support the persistence of multiple projects. At project creation, the database where the project's data will be persisted is to be chosen.

Unregister

A database used for storage cannot be unregistered if there are still projects linked to it. If this is the case, remove or archive the corresponding projects and then unregister the database (any remaining data will be untouched).

Edit

Limited edition of the database is possible when a database is in production.

Test

Opal server reports the result of a connection attempt. This allows to validate the connection url and credentials. This does not verifies that the database permissions are appropriate for the declared usage.

Plugins Administration

Overview

Summary

Plugins can be managed from the administration page:

- installed plugins
- plugins that can be upgraded
- new plugins that can be installed
- plugin manual installation
- plugins repository reference

Installed

The installed plugins are listed. Some operations can be performed on each plugin:

- a plugin is executed as a service which can be restarted.
- a plugin can be configured by editing the plugin's `site.properties` file. Depending on the plugin installation it can be necessary to restart the plugin so that the new configuration become effective.
- a plugin can be removed: it is in fact marked as being ready for removal and is still operational until the next Opal restart.

Updates

The plugin repository is inspected to list if some installed plugins have a most recent version available for install (according to the current Opal version).

Available

The plugin repository is inspected to list the plugins that are not installed and are available for installation (according to the current Opal version).

Advanced

Plugin Archive Installation

It is possible to install manually a plugin from its archive distribution. User is responsible for ensuring that the plugin applies to the current Opal version. The installation is effective at Opal restart.

Update Site

A plugin repository can be configured so that Opal can query the plugin updates and availability for installation. See Plugins Configuration instructions.

Java Virtual Machine Monitoring

Overview

- Summary
- Basic Monitoring
- New Relic Monitoring

Summary

This section is for monitoring the state of the Opal server.

Opal offers basic JVM monitoring. If you need more powerful metrics, consider using New Relic free services.

Basic Monitoring

This page gives you information about:

- Java running Opal:
    - Java version
    - VM name
    - VM vendor
    - VM version
- State of the Java Virtual Machine:
    - Heap and Non-Heap memory
    - Number of threads
    - Garbage Collector status
- System properties

New Relic Monitoring

See Opal Monitoring Configuration Guide.

My Profile

Overview

Introduction

Account Settings

Users can change their account password. For security reasons, the new password must be at least 6 characters long.

Bookmarks

In Opal three types of resources can be bookmarked by a user:

- Projects
- Tables
- Variables

Add a Bookmark

To bookmark a resource the user simply clicks on the star icon in the resource title. A yellow icon implies that the resource has been bookmarked.

View Bookmarks

There are two locations where the bookmarks are listed: in the user's **My Profile** section and in Opal's **Dashboard** page. To go to a resource section, user can click on the resource URL in the table.

Delete a bookmark

Bookmarks can only be removed from the user's **My Profile** section.

Opal Python User Guide

Contents of this Guide

Summary

The Opal Python Client is a command-line tool allowing access to an Opal server through its REST API interface. This is the ideal tool for automating Opal data, meta-data and configuration management, or it can be used in a custom Python application.

Installation

There are two methods to install a Opal Python Client package:

- use a Python package
- use the Debian package manager

Please read Opal Python Installation Guide for more details.

Usage

To get the options of the command line:

```
opal --help
```

This command will display which sub-commands are available. For each sub-command you can get the help message as well:

```
opal <subcommand> --help
```

Commands

Datasources Commands

These commands allow to access to both variables and values.

Command	Description
dict	Query for data dictionnary.
data	Query for data.
entity	Query for entities (Participant, etc.).
copy-table	Copy one or more tables into another or same project.
delete-table	Delete some tables of a project.
export-annot	Export the annotations that were applied to variables. Can be used to backup the annotations.
import-annot	Import the variable annotations. Can be used to restore annotations.

Import Commands

Import data from files (CSV, XML, SPSS) or a remote server (Opal, Limesurvey).

Command	Description
import-csv	Import data from a CSV file.
import-xml	Import data from a zip of XML files.
import-spss	Import data from a SPSS file.
import-r-spss	Import data from a SPSS file using R.
import-r-sas	Import data from a SAS file using R.
import-r-stata	Import data from a Stata file using R.

import-opal	Import data from a remote Opal server.
import-limesurvey	Import data from a LimeSurvey database.
import-sql	Import data from a SQL database.
import-ids	Import system identifiers from keyboard input or from a text file.
import-ids-map	Import identifiers mapping from keyboard input or from a text file.
import-vcf	Import some VCF/BCF files.

Export Commands

Export one or more tables to the Opal file system.

Command	Description
export-csv	Export data to a folder of CSV files.
export-xml	Export data to a zip of Opal XML files.
export-r-spss	Export data to a SPSS file.
export-r-sas	Export data to a SAS file.
export-r-stata	Export data to a Stata file.
export-sql	Export data to a SQL database.
export-vcf	Export some VCF/BCF files.

User and Group Commands

These commands allow to add, update or remove users and groups.

Command	Description
user	Manage users.
group	Manage groups.

Permission Commands

These commands allow to set/remove permissions on projects, tables, variables, DataSHIELD etc.

Command	Description
perm-project	Set/remove project related permissions.
perm-datasource	Set/remove datasource (the container of tables) related permissions.
perm-table	Set/remove table related permissions.
perm-variable	Set/remove variable related permissions.
perm-system	Set/remove system related permissions.
perm-datashield	Set/remove DataSHIELD service related permissions.
perm-r	Set/remove R service related permissions.

Other Commands

Command	Description

file	File system management.
system	Query for system status and configuration.
plugin	Plugin management.
rest	Request directly the Opal REST API (for advanced users).

Opal Python Installation Guide

Contents of this Guide

Introduction

There are two ways to install the Opal Python Client package on your computer:

- either from the Debian package (Debian-base Linux only),
- or from the Python package (Linux or Windows).

Installing Opal Python Client

Installation of the Debian package (recommended)

Opal Python client is available as a Debian package from OBiBa Debian repository.

Download Opal Python Client Debian package

Then Install the package:

```
sudo apt-get install opal-python-client
```

Some Debian-based systems (Debian Wheezy or Ubuntu Precise releases) only provide an older version of *python-protobuf* package dependency. When executing the opal python client the following error is reported:

```
ImportError: cannot import name enum_type_wrapper
```

This can be fixed manually using the commands:

```
apt-get purge python-protobuf

apt-get install python-pip

pip install protobuf>=2.5.0

apt-get install opal-python-client
```

Installation of the RPM package (recommended for Fedora-based systems)

Opal Python client is available as a RPM package from OBiBa RPM repository.

Download Opal Python Client RPM package

Then Install the package:

```
sudo yum install opal-python-client
```

Installation of the Python package

This type of package is cross-platform (Linux, Windows, Mac).

Install on Linux or Mac

1. Download the most recent version from:

   Download Opal Python Client package
2. Decompress the file and enter the installation folder:

   ```
   tar xvzf opal-python-client-X.XX.tar.gz
   cd opal-python-client-X.XX
   ```

3. Install the package:

   ```
   sudo python setup.py install --record installed_files.lst
   ```

   > The '–record' will generate a list of installed files on your system. Since there is no uninstaller, you can use this file to remove the Opal Python Client package. You can do this by executing the following command:
   >
   > ```
   > $ sudo cat installed_files.lst | xargs rm -rf
   > ```

Install on Windows

Using Cygwin

You can install Cygwin, making sure that CURL, Python, gcc are included and follow these steps inside a Cygwin BASH window:

```
cd /usr/lib
cp libcurl.dll.a libcurl.a
cd <your-desired-dir>
curl -C - -O
http://download.obiba.org/opal/stable/opal-python-client-X.XX.tar.gz
tar xzvf opal-python-client-X.XX.tar.gz
cd opal-python-client-X.XX
python setup.py install --record installed_files.lst
```

Using plain Windows tools

This Windows installation is the most complicated one but does not required any third party tools. You are required to do a few manual installations before the package is fully usable. The following steps were tested on a Windows 7.

1. You must have Python installed on your Windows system. Run this installer in case you don't have one.

2. Download the Google protobuf binary and make sure that its containing folder is in your path.
3. Download the protobuf source package containing the setup.py file and follow these steps:

```
unzip protobuf-2.5.0.zip
cd protobuf-2.5.0/python
python setup.py install
```

4. Go to the Python Libs site and download the file ***pycurl-7.19.0.win-amd64-py2.7.exe***
5. Run the installer and follow the instructions until the package is installed
6. Download the http://download.obiba.org/opal/stable/opal-python-client-X.XX.zip and follow these steps:

```
unzip
http://download.obiba.org/opal/stable/opal-python-client-X.XX.zip
cd opal-python-client-X.XX
python setup.py bdist_wininst
cd dist
```

7. Execute the generated installer and follow the instructions (opal-python-client-X.XX.win-amd64.exe)

Testing the Installation

Test the installation by performing a REST call to your Opal server:

```
opal rest /datasources --method GET  --accept application/json   --opal
http://<YOUR-OPAL-SERVER-IP>:8080  --user <YOUR-USER> --password
<YOUR-PASSWORD> --json
```

Opal Python Commands

- Datasources Commands
- Import Commands
- Export Commands
- User and Group Commands
- Permission Commands
- Other Commands

Datasources Commands

- Data Dictionnary Command
- Data Command
- Entities Command
- Table Copy Command
- Table Delete Command
- Export Annotations Command
- Import Annotations Command

Data Dictionnary Command

Contents of this page

- Synopsis
- Examples

Synopsis

```
opal dict <datasource | datasource.table | datasource.table:variable>
<CREDENTIALS> [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

To fetch the dictionnary associated to a datasource:

```
opal dict mica_demo --opal https://opal-demo.obiba.org --user administrator
--password password
```

To fetch the dictionnary associated to a table in a pretty format:

```
 opal dict mica_demo.FNAC --opal https://opal-demo.obiba.org --user
administrator --password password -j
```

To fetch the description of a variable:

```
opal dict mica_demo.HOP:PM_BMI_CONTINUOUS -o https://opal-demo.obiba.org -u
administrator -p password -j
```

Wild cards can also be used:

```
# Get all datasources
opal dict "*" --opal https://opal-demo.obiba.org --user administrator
--password password

# Get all tables from mica_demo datasource
opal dict "mica_demo.*" --opal https://opal-demo.obiba.org --user
administrator --password password

# Get all variables from mica_demo.FNAC table
opal dict "mica_demo.FNAC:*" --opal https://opal-demo.obiba.org --user
administrator --password password
```

Data Command

Contents of this page

Synopsis

```
opal data <datasource.table | datasource.table:variable> <CREDENTIALS>
[OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--id ID, -i ID	Entity identifier. If missing the list of entities is returned
--raw, -r	Raw value
--pos POS, -po POS	Position of the value to query in case of a repeatable variable (starting at 0).

Extras

Options	Description
--help, -h	Displays the command's help message

--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Fetch the list of entities associated to a table:

```
opal data mica_demo.FNAC --opal https://opal-demo.obiba.org --user
administrator --password password
```

Fetch the variable value for a entity in a table:

```
# Get the JSON representation of the value
opal data mica_demo.FNAC:SUKUP -o https://opal-demo.obiba.org -u
administrator -p password --id 1030243378182074458 -j

# Get the raw value. If variable is of binary type, a byte stream is
outputed.
opal data mica_demo.FNAC:SUKUP -o https://opal-demo.obiba.org -u
administrator -p password --id 1030243378182074458 -j --raw
```

Entities Command

Contents of this page

Synopsis

```
opal entity <id> <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Id

Id is the identifier of the entity.

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--type TYPE, -ty TYPE	Type of the entity (e.g. Participant, Instrument, Drug). Default type is Participant.
--tables, -ta	To get the list of tables in which the entity with given id exists.

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Fetch the entities with id 1030243378182074458:

```
opal entity 1030243378182074458 --opal https://opal-demo.obiba.org --user
administrator --password password
```

Fetch the list of table where entity 1030243378182074458 exists:

```
opal entity 1030243378182074458 --opal https://opal-demo.obiba.org --user
administrator --password password --tables
```

Fetch the list of table where entity 1030243378182074458 of type "Participant" exists:

```
opal entity 1030243378182074458 --opal https://opal-demo.obiba.org --user
administrator --password password --tables --type Participant
```

Table Copy Command

Contents of this page

Synopsis

```
opal copy-table <CREDENTIALS> [OPTIONS] [XTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal

--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--project, -pr	Source project name
--tables, -t	List of table names which will be copied (default is all)
--destination, -d	Destination project name
--name, -na	New table name (required if source and destination are the same, ignored if more than one table is to be copied)
--incremental, -i	Incremental copy
--nulls, -nu	Copy the null values

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Copy a table in the same project, by specifying a new name:

```
opal copy-table --opal https://opal-demo.obiba.org --user administrator
--password password --project mica_demo --tables HOP  --destination
mica_demo --name HOP2
```

If the name of a project, table or subject contains blanks, enclose it with quotes:

```
... --name "HOP 2" ...
```

Table Delete Command

Contents of this page

Synopsis

```
opal delete-table <CREDENTIALS> [OPTIONS] [XTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--project, -pr	Source project name
--tables, -t	List of table names which will be deleted (default is all)

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command

Examples

Delete some tables from a project:

```
opal delete-table --opal https://opal-demo.obiba.org --user administrator
--password password --project project_test --tables Table1 Table2
```

Delete all tables from a project:

```
opal delete-table --opal https://opal-demo.obiba.org --user administrator
--password password --project project_test
```

If the name of a project, table or subject contains blanks, enclose it with quotes:

```
... --tables "Table 1" ...
```

Export Annotations Command

- Synopsis
- Examples

Synopsis

```
opal export-annot <datasource | datasource.table> <CREDENTIALS> [OPTIONS]
[EXTRAS]
```

Credentials

Options	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Options	Description
--output OUTPUT, -out OUTPUT	CSV/TSV file path where to write the exported annotations. When not specified, standard output is used.
--separator SEPARATOR, -s SEPARATOR	The character separator to be used in the output. When not specified tab character is used.
--taxonomies TAXONOMIES [TAXONOMIES ...], -tx TAXONOMIES [TAXONOMIES ...]	The list of taxonomy names of interest (default is any that are found in the variable attributes).

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command

Examples

Export annotations of the `Mlstr_area` taxonomy from a specific table:

```
opal export-annot --opal https://opal-demo.obiba.org --user administrator
--password password Study1.DatasetA --taxonomies Mlstr_area --out
/tmp/study1-datasetA-area.tsv
```

Import Annotations Command

Contents of this page

Synopsis

```
opal import-annot <CREDENTIALS> <DESTINATION> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--input INPUT, -in INPUT	CSV/TSV input file, typically the output of the export-annot command (default is stdin)
--separator SEPARATOR, -s SEPARATOR	Separator char for CSV/TSV format (default is the tabulation character)
--destination DESTINATION, -d DESTINATION	Destination datasource name (default is the one(s) specified in the input file)
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables which variables are to be annotated (defaults to all that are found in the input file)
--taxonomies TAXONOMIES [TAXONOMIES ...], -tx TAXONOMIES [TAXONOMIES ...]	The list of taxonomy names of interest (default is any that is found in the input file)

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Import some annotations to a specified table:

```
opal import-annot --user administrator --password password --destination
Study2 --tables datasetA --input /tmp/area-annotations.tsv
```

Import Commands

Import CSV Command

Contents of this page

Synopsis

```
opal import-csv <CREDENTIALS> <DESTINATION> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Destination

Option	Description
--destination DESTINATION, -d DESTINATION	Destination datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be imported (defaults to all)
--incremental -i	Incremental import
--identifiers IDENTIFIERS, -id IDENTIFIERS	Name of the ID mapping
--policy POLICY, -po POLICY	ID mapping policy: • *required* : each identifiers must be mapped prior importation (default), • *ignore* : ignore unknown identifiers, • *generate* : generate a system identifier for each unknown identifier.

Options

Option	Description
--path PATH -pa PATH	Path to the CSV file to import on the Opal file system
--characterSet CHARACTERSET, -c CHARACTERSET	Character set of the file (e.g utf-8)
--separator SEPARATOR, -s SEPARATOR	Field separator
--quote QUOTE, -q QUOTE	Quotation mark character
--firstRow FIRSTROW, -f FIRSTROW	Number of the first row that contains data to import
--type TYPE, -ty TYPE	Entity type of the data (e.g. Participant)

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Import catchment areas from a csv file delimited with ',' :

```
opal import-csv --opal https://opal-demo.obiba.org --user administrator
--password password --destination opal-data --path
/home/administrator/catchment-area.csv --tables catchment-area --separator
, --type Area
```

Import XML Command

Contents of this page

Synopsis

```
opal import-xml <CREDENTIALS> <DESTINATION> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file

--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Destination

Option	Description
--destination DESTINATION, -d DESTINATION	Destination datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be imported (defaults to all)
--incremental -i	Incremental import
--identifiers IDENTIFIERS, -id IDENTIFIERS	Name of the ID mapping
--policy POLICY, -po POLICY	ID mapping policy: • *required* : each identifiers must be mapped prior importation (default), • *ignore* : ignore unknown identifiers, • *generate* : generate a system identifier for each unknown identifier.

Options

Option	Description
--path PATH -pa PATH	Path to the zip of XML files to import on the Opal file system

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Import tables from 20-onyx ZIP file to the opal-data datasource:

```
# Import all tables
opal import-xml --opal https://opal-demo.obiba.org --user administrator
--password password --path /home/administrator/20-onyx-data.zip
--destination opal-data

# Import only ArmSpan and BloodPressure tables
opal import-xml --opal https://opal-demo.obiba.org --user administrator
--password password --path /home/administrator/20-onyx-data.zip
--destination opal-data --tables ArmSpan BloodPressure
```

Import SPSS Command

Contents of this page

Synopsis

```
opal import-spss <CREDENTIALS> <DESTINATION> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Destination

Option	Description
--destination DESTINATION, -d DESTINATION	Destination datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be imported (defaults to all)
--incremental -i	Incremental import
--identifiers IDENTIFIERS, -id IDENTIFIERS	Name of the ID mapping
--policy POLICY, -po POLICY	ID mapping policy: • *required* : each identifiers must be mapped prior importation (default), • *ignore* : ignore unknown identifiers, • *generate* : generate a system identifier for each unknown identifier.

Options

Option	Description
--path PATH -pa PATH	Path to the SPSS file to import on the Opal file system
--locale LOCALE, -l LOCALE	Language code to be associated to labels
--characterSet CHARACTERSET, -c CHARACTERSET	Character set to be used when reading the file
--type TYPE, -ty TYPE	Entity type (default is Participant)
--idVariable IDVARIABLE, -iv IDVARIABLE	SPSS variable that provides the entity ID. If not specified, first variable values are considered to be the entity identifiers.

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Import table from RobotChicken SPSS file in opal-data datasource:

```
opal import-spss --opal https://opal-demo.obiba.org --user administrator
--password password --destination opal-data --characterSet ISO-8859-1
--locale en --path /home/administrator/RobotChicken.sav
```

Import SAS (using R) Command

Contents of this page

Synopsis

```
opal import-r-sas <CREDENTIALS> <DESTINATION> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Destination

Option	Description
--destination DESTINATION, -d DESTINATION	Destination datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be imported (defaults to all)
--incremental -i	Incremental import
--identifiers IDENTIFIERS, -id IDENTIFIERS	Name of the ID mapping
--policy POLICY, -po POLICY	ID mapping policy: • *required* : each identifiers must be mapped prior importation (default), • *ignore* : ignore unknown identifiers, • *generate* : generate a system identifier for each unknown identifier.

Options

Option	Description
--path PATH, -pa PATH	Path to the SAS file to import on the Opal file system

--locale LOCALE, -l LOCALE	Language code to be associated to the labels
--type TYPE, -ty TYPE	Entity type (default is Participant)
--idVariable IDVARIABLE, -iv IDVARIABLE	SAS variable that provides the entity ID. If not specified, first variable values are considered to be the entity identifiers.

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Import table from RobotChicken SAS file in opal-data datasource:

```
opal import-r-sas --opal https://opal-demo.obiba.org --user administrator
--password password --destination opal-data --locale en --path
/home/administrator/RobotChicken.sas7bdat
```

Import SPSS (using R) Command

Contents of this page

Synopsis

```
opal import-r-spss <CREDENTIALS> <DESTINATION> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Destination

Option	Description
--destination DESTINATION, -d DESTINATION	Destination datasource name

--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be imported (defaults to all)
--incremental -i	Incremental import
--identifiers IDENTIFIERS, -id IDENTIFIERS	Name of the ID mapping
--policy POLICY, -po POLICY	ID mapping policy: • *required* : each identifiers must be mapped prior importation (default), • *ignore* : ignore unknown identifiers, • *generate* : generate a system identifier for each unknown identifier.

Options

Option	Description
--path PATH, -pa PATH	Path to the SPSS file to import on the Opal file system
--locale LOCALE, -l LOCALE	Language code to be associated to the labels
--type TYPE, -ty TYPE	Entity type (default is Participant)
--idVariable IDVARIABLE, -iv IDVARIABLE	SPSS variable that provides the entity ID. If not specified, first variable values are considered to be the entity identifiers.

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Import table from RobotChicken SPSS file in opal-data datasource:

```
opal import-r-spss --opal https://opal-demo.obiba.org --user administrator
--password password --destination opal-data --locale en --path
/home/administrator/RobotChicken.sav
```

Import Stata (using R) Command

Contents of this page

Synopsis

```
opal import-r-stata <CREDENTIALS> <DESTINATION> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Destination

Option	Description
--destination DESTINATION, -d DESTINATION	Destination datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be imported (defaults to all)
--incremental -i	Incremental import
--identifiers IDENTIFIERS, -id IDENTIFIERS	Name of the ID mapping
--policy POLICY, -po POLICY	ID mapping policy: • *required* : each identifiers must be mapped prior importation (default), • *ignore* : ignore unknown identifiers, • *generate* : generate a system identifier for each unknown identifier.

Options

Option	Description
--path PATH, -pa PATH	Path to the Stata file to import on the Opal file system
--locale LOCALE, -l LOCALE	Language code to be associated to the labels
--type TYPE, -ty TYPE	Entity type (default is Participant)
--idVariable IDVARIABLE, -iv IDVARIABLE	Stata variable that provides the entity ID. If not specified, first variable values are considered to be the entity identifiers.

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Import table from RobotChicken Stata file in opal-data datasource:

```
opal import-r-stata --opal https://opal-demo.obiba.org --user administrator
--password password --destination opal-data --locale en --path
/home/administrator/RobotChicken.dta
```

Import Opal Command

Contents of this page

Synopsis

```
opal import-opal <CREDENTIALS> <REMOTE_CREDENTIALS> <DESTINATION> [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Remote Credentials

Option	Description
--ropal ROPAL, -ro ROPAL	Remote Opal server base url
--ruser RUSER, -ru RUSER	Remote User name to connect to Opal
--rpassword RPASSWORD, -rp RPASSWORD	Remote User's password
--key KEY, -k KEY	Location of the certificate key

Destination

Option	Description
--rdatasource RDATASOURCE, -rd RDATASOURCE	Remote datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be imported from the remote datasource (defaults to all)
--destination DESTINATION, -d DESTINATION	Destination datasource name (into which data will be imported)
--incremental -i	Incremental import
--identifiers IDENTIFIERS, -id IDENTIFIERS	Name of the ID mapping
--policy POLICY, -po POLICY	ID mapping policy: • *required* : each identifiers must be mapped prior importation (default), • *ignore* : ignore unknown identifiers, • *generate* : generate a system identifier for each unknown identifier.

Extras

Options	Description

--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Copy tables BloodPressure and ArmSpan from Opal on demo.obiba.org to Opal on localhost:

```
opal import-opal -o http://localhost:8080 -u administrator -p password
--ro https://opal-demo.obiba.org --ru administrator --rp password
--rdatasource onyx --destination opal-data --tables BloodPressure ArmSpan
```

Import LimeSurvey Command

Contents of this page

Synopsis

```
opal import-limesurvey <CREDENTIALS> <DESTINATION> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Destination

Option	Description
--destination DESTINATION, -d DESTINATION	Destination datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be imported (defaults to all)
--incremental -i	Incremental import
--identifiers IDENTIFIERS, -id IDENTIFIERS	Name of the ID mapping
--policy POLICY, -po POLICY	ID mapping policy: • *required* : each identifiers must be mapped prior importation (default), • *ignore* : ignore unknown identifiers, • *generate* : generate a system identifier for each unknown identifier.

Options

Option	Description
--database DATABASE, -db DATABASE	Name of the LimeSurvey SQL database as registered in Opal
--prefix PREFIX -pr PREFIX	Table prefix of LimeSurvey tables (default: none)

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Import the table "Withdrawal Script (WaveInter-wave contact)" from LimeSurvey database to the opal-data datasource:

```
opal import-limesurvey --opal https://opal-demo.obiba.org --user
administrator --password password --destination ds1 --database LimeSurvey
--json -t "Withdrawal Script (WaveInter-wave contact)"
```

Import SQL Command

Contents of this page

Synopsis

```
opal import-sql <CREDENTIALS> <DESTINATION> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Destination

Option	Description

--destination DESTINATION, -d DESTINATION	Destination datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be imported (defaults to all)
--incremental -i	Incremental import
--identifiers IDENTIFIERS, -id IDENTIFIERS	Name of the ID mapping
--policy POLICY, -po POLICY	ID mapping policy: • *required* : each identifiers must be mapped prior importation (default), • *ignore* : ignore unknown identifiers, • *generate* : generate a system identifier for each unknown identifier.

Options

Option	Description
--database DATABASE, -db DATABASE	Name of the SQL database as registered in Opal

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Import the table "AnkleBrachial" from a SQL database to the opal-data datasource:

```
opal import-sql --opal https://opal-demo.obiba.org --user administrator
--password password --destination ds1 --database sql_db --json -t
AnkleBrachial
```

Import System Identifiers Command

Contents of this page

Synopsis

```
opal import-ids <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url

--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--type TYPE, -t TYPE	Entity type of the data (e.g. Participant)

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command

Examples

Import system identifiers from keyboard entries.

```
opal import-ids --opal https://opal-demo.obiba.org --user administrator
--password password --type Participant
```

Import system identifiers from a file.

```
opal import-ids --opal https://opal-demo.obiba.org --user administrator
--password password --type Participant < ids.txt
```

Example of a file of identifiers:

```
11123456
11345467
11995884
11423423
```

Import Identifiers Mapping Command

Contents of this page

Synopsis

```
opal import-ids-map <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--type TYPE, -t TYPE	Entity type of the data (e.g. Participant)
--map MAP, -m MAP	Mapping name, i.e. the name associated to the identifiers that will be mapped to the system identifiers
--separator SEP, -s SEP	Identifiers separator (default is ",")

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command

Examples

Import identifiers mapping from keyboard entries.

```
opal import-ids-map --opal https://opal-demo.obiba.org --user administrator
--password password --type Participant --map foo
```

Import identifiers mapping from a file.

```
opal import-ids-map --opal https://opal-demo.obiba.org --user administrator
--password password --type Participant --map foo < idsmap.txt
```

Example of a file of identifiers mapping:

```
11123456,A11111
11345467,A22222
11995884,A33333
11423423,A44444
```

Import VCF Command

- Synopsis

- Examples

Synopsis

```
opal import-vcf <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Options	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Options	Description
--project PROJECT, -pr PROJECT	Project name into which genotypes data will be imported
--vcf FILE01 FILE02, -vcf FILE01 FILE02	List of VCF/BCF (optionally compressed) file paths (in Opal file system)

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command

Examples

Import VCF files into the project TEST:

```
opal import-vcf --opal https://opal-demo.obiba.org --user administrator
--password password --project TEST --vcf /path/to/file01.vcf.gz
/path/to/file02.vcf.gz
```

Export Commands

- Export CSV Command
- Export XML Command
- Export SPSS (using R) Command
- Export SAS (using R) Command
- Export Stata (using R) Command
- Export SQL Command
- Export VCF Commands

Export CSV Command

Contents of this page

Synopsis

```
opal export-csv <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--datasource DATASOURCE, -d DATASOURCE	Datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be exported
--incremental -i	Incremental export
--identifiers ID_MAPPING, -id ID_MAPPING	Entity ID mapping name
--no-multilines, -nl	Do not write value sequences as multiple lines
--output OUTPUT, -out OUTPUT	Output directory name on the Opal file system

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Export tables from opal-data:

```
# Export the Waist table into /home/administrator. A subdirectory Waist is
created with table definition and data as csv files.
opal export-csv --opal https://opal-demo.obiba.org --user administrator
--password password --datasource opal-data --tables BloodPressure --output
/tmp/export
```

Export XML Command

Contents of this page

- Synopsis
- Examples

Synopsis

```
opal export-xml <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--datasource DATASOURCE, -d DATASOURCE	Datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be exported
--incremental -i	Incremental export
--identifiers ID_MAPPING, -id ID_MAPPING	Entity ID mapping name
--output OUTPUT, -out OUTPUT	Output file name on the Opal file system

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Export tables from opal-data to a ZIP file of XML files:

```
# Export the Spirometry and StandingHeight tables
opal export-xml --opal https://opal-demo.obiba.org --user administrator
--password password --datasource opal-data --tables Spirometry
StandingHeight --output /tmp/export.zip
```

Export SPSS (using R) Command

Contents of this page

- Synopsis
- Examples

Synopsis

```
opal export-r-spss <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--datasource DATASOURCE, -d DATASOURCE	Datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be exported
--identifiers ID_MAPPING, -id ID_MAPPING	Entity ID mapping name
--output OUTPUT, -out OUTPUT	Output SPSS file name on the Opal file system

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Export table from opal-data to a SPSS file:

```
opal export-r-spss --opal https://opal-demo.obiba.org --user administrator
--password password --datasource opal-data --tables StandingHeight --output
/tmp/sh.sav
```

Export SAS (using R) Command

Contents of this page

Synopsis

```
opal export-r-sas <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--datasource DATASOURCE, -d DATASOURCE	Datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be exported
--identifiers ID_MAPPING, -id ID_MAPPING	Entity ID mapping name
--output OUTPUT, -out OUTPUT	Output SAS file name on the Opal file system

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Export table from opal-data to a SAS file:

```
opal export-r-sas --opal https://opal-demo.obiba.org --user administrator
--password password --datasource opal-data --tables StandingHeight --output
/tmp/sh.sas7bdat
```

Export Stata (using R) Command

Contents of this page

Synopsis

```
opal export-r-stata <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--datasource DATASOURCE, -d DATASOURCE	Datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be exported
--identifiers ID_MAPPING, -id ID_MAPPING	Entity ID mapping name
--output OUTPUT, -out OUTPUT	Output Stata file name on the Opal file system

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Export table from opal-data to a Stata file:

```
opal export-r-stata --opal https://opal-demo.obiba.org --user administrator
--password password --datasource opal-data --tables StandingHeight --output
/tmp/sh.dta
```

Export SQL Command

Contents of this page

Synopsis

```
opal export-sql <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--datasource DATASOURCE, -d DATASOURCE	Datasource name
--tables TABLES [TABLES ...], -t TABLES [TABLES ...]	The list of tables to be exported
--incremental -i	Incremental export
--identifiers ID_MAPPING, -id ID_MAPPING	Entity ID mapping name
--database DATABASE, -db DATABASE	Name of the SQL database as registered in Opal

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Export tables from opal-data to a SQL database:

```
# Export the Spirometry table into /home/administrator.
opal export-xml --opal https://opal-demo.obiba.org --user administrator
--password password --datasource opal-data --tables Spirometry  --database
sql_db
```

Export VCF Commands

- Synopsis
- Examples

Synopsis

```
opal export-vcf <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Options	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Options	Description
--project PROJECT, -pr PROJECT	Project name from which genotypes data will be exported
--vcf FILE01 FILE02, -vcf FILE01 FILE02	List of VCF/BCF file names
--destination DESTINATION, -d DESTINATION	Destination folder (in Opal file system)
--filter-table FILTER -f FILTER	Participant table name to be used to filter the samples by participant ID (only relevant if there is a sample-participant mapping defined)
--no-case-controls, -nocc	Do not include case control samples (only relevant if there is a sample-participant mapping defined)

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command

Examples

Export VCF files into the user's export directory, omitting control samples:

```
opal export-vcf --opal https://opal-demo.obiba.org --user administrator
--password password --project TEST --vcf FILE01 FILE02 --destination
/home/administrator/export --filter-table TEST.mapping --no-case-controls
```

User and Group Commands

- User Command
- Group Command

User Command

Contents of this page

- Synopsis
- Examples

Synopsis

```
opal user <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--name NAME, -n NAME	User name
--fetch, -f	Fetch a user (all users are returned if no option --name is given)
--delete, -de	Delete the user specified by the --name option
--add, -a	Add a user specified by the --name option
--update, -ud	Update user password/certificate, status (enable/disable) and groups
--upassword, -upa	User password of at least six characters.
--ucertificate, -uc	User certificate (public key) file
--disabled, -di	Disable user account (if omitted the user is enabled by default).
--groups, -g	User groups (separated by space)

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

```
# Get the list of users with associated groups
opal user --opal https://opal-demo.obiba.org --user administrator
--password password --fetch

# Get a specific user with associated groups
opal user --opal https://opal-demo.obiba.org --user administrator
--password password --fetch --name user1

# Create the user user2
opal user --opal https://opal-demo.obiba.org --user administrator
--password password --add --name user2 --upassword 123456

# Create the user user3 with a certificate
opal user --opal https://opal-demo.obiba.org --user administrator
--ucertificate /path/to/certifcate.pem --add --name user3

# Create the user (disabled) user4 with groups group1, group2 and group3
opal user --opal https://opal-demo.obiba.org --user administrator
--password password --add --name user4 --disabled --upassword 123456
--groups  group1 group2 group3

# Update user2's password
opal user --opal https://opal-demo.obiba.org --user administrator
--password password --update --name user2 --upassword 987654

# Update user2's status, set to disabled
opal user --opal https://opal-demo.obiba.org --user administrator
--password password --update --name user2 --disabled

# Update user2's status, set to enabled
opal user --opal https://opal-demo.obiba.org --user administrator
--password password --update --name user2

# Update user2's groups
opal user --opal https://opal-demo.obiba.org --user administrator
--password password --update --name user2 --groups group1 group2

# Delete the user user3
opal user --opal https://opal-demo.obiba.org --user administrator
--password password --delete --name user2
```

Group Command

Contents of this page

Synopsis

```
opal group <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--name NAME, -n NAME	Group name
--fetch, -f	Fetch a group (all groups are returned if no option --name is given)
--delete, -de	Delete the group with the name specified by the --name option

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

```
# Get the list of groups with associated users
opal group --opal https://opal-demo.obiba.org --user administrator
--password password --fetch

# Get a specific group with associated users
opal group --opal https://opal-demo.obiba.org --user administrator
--password password --fetch --name editor

# Delete the group study_editor
opal group --opal https://opal-demo.obiba.org --user administrator
--password password --delete --name study_editor
```

Permission Commands

- Project Permission Command

- Datasource Permission Command
- Table Permission Command
- Variable Permission Command
- System Permission Command
- DataSHIELD Permission Command
- R Permission Command

Project Permission Command

Contents of this page

Synopsis

```
opal perm-project <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--add, -a	Add a permission
--delete, -d	Delete a permission
--permission, -pe	Permission to apply (administrate)
--subject, -s	Subject name to which the permission will be granted
--type TYPE, -ty TYPE	Subject type: user or group
--project, -pr	Project name on which the permission is to be set

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Add **administrate** permission for subject **demouser** on **mica_demo** project:

```
opal perm-project --opal https://opal-demo.obiba.org --user administrator
--password password --type USER --subject demouser --permission
administrate --project mica_demo --add
```

Remove the above permission:

```
opal perm-project --opal https://opal-demo.obiba.org --user administrator
--password password --type USER --subject demouser --project mica_demo
--delete
```

> If the name of a project or subject contains blanks, enclose it with quotes:
>
> ```
> ... --project "mica demo" --subject "demo user" --permission view
> --add
> ```

Datasource Permission Command

Contents of this page

- Synopsis
- Examples

Synopsis

```
opal perm-datasource <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--add, -a	Add a permission
--delete, -d	Delete a permission

--permission, -pe	Permission to apply (refer to Permissions table for more info)
--subject, -s	Subject name to which the permission will be granted
--type TYPE, -ty TYPE	Subject type: user or group
--project, -pr	Project name to which the datasource belongs

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Permissions

Option	Description
view-value	View dictionary and values of all tables
add-table	Add tables or views
administrate	Administrate

Examples

Add **add-table** permission for subject **demouser** on **mica_demo** project:

```
opal perm-datasource --opal https://opal-demo.obiba.org --user
administrator --password password --type USER --subject demouser
--permission add-table --project mica_demo --add
```

Remove the above permission:

```
opal perm-datasource --opal https://opal-demo.obiba.org --user
administrator --password password --type USER --subject demouser  --project
mica_demo --delete
```

If the name of a project or subject contains blanks, enclose it with quotes:

```
... --project "mica demo" --subject "demo user" --permission view
--add
```

Table Permission Command

Contents of this page

- Examples

Synopsis

```
opal perm-table <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--add, -a	Add a permission
--delete, -d	Delete a permission
--permission, -pe	Permission to apply (refer to Permissions table for more info)
--subject, -s	Subject name to which the permission will be granted
--type TYPE, -ty TYPE	Subject type: user or group
--project, -pr	Project name to which the tables belong
--tables, -t	List of table names on which the permission is to be set (default is all)

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Permissions

Option	Description
view	View dictionary and summary
view-value	View dictionary and values
edit	Edit dictionary and view summary
edit-values	Edit dictionary and view values

administrate	Administrate

Examples

Add **view** permission for subject **demouser** on table **HOP** in **mica_demo** project:

```
opal perm-table --opal https://opal-demo.obiba.org --user administrator
--password password --type USER --project mica_demo --subject demouser
--permission view --add --tables HOP
```

Remove the above permission:

```
opal perm-table --opal https://opal-demo.obiba.org --user administrator
--password password --type USER --project mica_demo --subject demouser
--delete --table HOP
```

Add permission on all tables of **mica_demo** project:

```
opal perm-table --opal https://opal-demo.obiba.org --user administrator
--password password --type USER --project mica_demo --subject demouser
--permission view --add
```

Remove permission from all table of **mica_demo** project:

```
opal perm-table --opal https://opal-demo.obiba.org --user administrator
--password password --type USER --project mica_demo --subject demouser
--delete
```

Add permission on specific tables:

```
opal perm-table --opal https://opal-demo.obiba.org --user administrator
--password password --type USER --project mica_demo --subject demouser
--permission view --add --tables HOP FNAC
```

> If the name of a project, table or subject contains blanks, enclose it with quotes:
>
> ```
> ... --project "mica demo" --subject "demo user" --permission view
> --add --tables "HOP 2"
> ```

Variable Permission Command

Contents of this page

Synopsis

```
opal perm-variable <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--add, -a	Add a permission
--delete, -d	Delete a permission
--permission, -pe	Permission to apply (view)
--subject, -s	Subject name to which the permission will be granted
--type TYPE, -ty TYPE	Subject type: user or group
--project, -pr	Project name to which the tables belong
--table, -t	Table name to which the variable belongs
--variables, -va	List of variable names on which the permission is to be set

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Add **view** permission for subject **demouser** on variables **SVUOSI** and **SUKUP** of table **HOP** in **mica_demo** project:

```
opal perm-variable --opal https://opal-demo.obiba.org --user administrator
--password password --type USER --subject demouser --project mica_demo
--table FNAC --variables SVUOSI SUKUP --permission view  --add
```

Remove the above permission:

```
opal perm-variable --opal https://opal-demo.obiba.org --user administrator
--password password --type USER --subject demouser --project mica_demo
--table FNAC --variables SVUOSI SUKUP --delete
```

If the name of a project, table, variable or a subject contains blanks, enclose it with quotes:

```
... --project "mica demo" --subject "demo user" --table "FNAC 2"
--variables "Var One" "Var Two" --permission view --add -v
```

System Permission Command

Contents of this page

Synopsis

```
opal perm-system <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--add, -a	Add a permission
--delete, -d	Delete a permission
--permission, -pe	Permission to apply (add-project, administrate)
--subject, -s	Subject name to which the permission will be granted
--type TYPE, -ty TYPE	Subject type: user or group

Extras

Options	Description

--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Add **add-project** permission for subject **demouser**:

```
opal perm-system --opal https://opal-demo.obiba.org --user administrator
--password password --type USER --subject demouser --permission add-project
--add
```

Remove the above permission:

```
opal perm-system --opal https://opal-demo.obiba.org --user administrator
--password password --type USER --subject demouser --delete
```

> If the name of a project or subject contains blanks, enclose it with quotes:
>
> ```
> ... --subject "demo user" --permission view --add
> ```

DataSHIELD Permission Command

Contents of this page

Synopsis

```
opal perm-datashield <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--add, -a	Add a permission
--delete, -d	Delete a permission
--permission, -pe	Permission to apply (use, administrate)
--subject, -s	Subject name to which the permission will be granted
--type TYPE, -ty TYPE	Subject type: user or group

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Add **use** permission for subject **demouser**:

```
opal perm-datashield --opal https://opal-demo.obiba.org --user
administrator --password password --type USER --subject demouser
--permission use --add
```

Remove the above permission:

```
opal perm-datashield --opal https://opal-demo.obiba.org --user
administrator --password password --type USER --subject demouser --delete
```

> If the name of a project, table or subject contains blanks, enclose it with quotes:
>
> ```
> ... --subject "demo user" --permission use --add
> ```

R Permission Command

Contents of this page

Synopsis

```
opal perm-r <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--add, -a	Add a permission
--delete, -d	Delete a permission
--permission, -pe	Permission to apply (use)
--subject, -s	Subject name to which the permission will be granted
--type TYPE, -ty TYPE	Subject type: user or group

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Add **use** permission for subject **demouser**:

```
opal perm-r --opal https://opal-demo.obiba.org --user administrator
--password password --type USER --subject demouser --permission use --add
```

Remove the above permission:

```
opal perm-r --opal https://opal-demo.obiba.org --user administrator
--password password --type USER --subject demouser --add
```

If the name of a project, table or subject contains blanks, enclose it with quotes:

```
... --subject "demo user" --permission use --add
```

Other Commands

File System Command

Contents of this page

Synopsis

```
opal file PATH <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Path

The path on the Opal file system.

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--download, -dl	Download a file or a folder (as a zip file)
--download-password PASSWORD, -dlp PASSWORD	Password to encrypt the file content.
--upload UPLOAD, -up UPLOAD	Upload a local file to a folder in Opal file system
--delete, -dt	Delete a file on Opal file system
--force, -f	Skip confirmation

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command

--json, -j	Output pretty-print JSON

Examples

Upload a file to Opal file system:

```
opal file --opal https://opal-demo.obiba.org --user administrator
--password password -up /path/to/local/file /home/administrator
```

Download a folder (zip file) from Opal file system:

```
opal file --opal https://opal-demo.obiba.org --user administrator
--password password -dl /home/administrator/export/collected >
collected.zip
```

Download a file from Opal file system:

```
opal file --opal https://opal-demo.obiba.org --user administrator
--password password -dl /home/administrator/HOP-FNAC2.xml > HOP-FNAC2.xml
```

Delete a file from Opal file system:

```
opal file --opal https://opal-demo.obiba.org --user administrator
--password password -dt /home/administrator/HOP-FNAC2.xml
```

System Command

Content of this page

Synopsis

```
opal system <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password

--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--conf	Opal application configuration (default option)
--version	Opal version number
--status	Opal application status (JVM related dynamic properties)

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Retrieve Opal configuration information:

```
opal system --opal https://opal-demo.obiba.org --user administrator
--password password --conf
```

Retrieve Opal version number:

```
opal system --opal https://opal-demo.obiba.org --user administrator
--password password --version
```

Retrieve Opal status and JVM related dynamic properties:

```
opal system --opal https://opal-demo.obiba.org --user administrator
--password password --status
```

Retrieve Opal java execution environment its JVM relates statistics properties:

```
opal system --opal https://opal-demo.obiba.org --user administrator
--password password --env
```

Plugin Command

Contents of this page

Synopsis

```
opal plugin <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
--list, -ls	List the installed plugins
--updates, -lu	List the installed plugins that can be updated
--available, -la	List the new plugins that could be installed.
--install NAME, -i NAME	Install a plugin by providing its `name` or `name:version` or a `path` to a plugin archive file (in Opal file system, ending with "`-dist.zip`"). If no version is specified, the latest version is installed. Requires system restart to be effective.
--remove NAMME, -rm NAME	Remove a plugin by providing its name. Requires system restart to be effective.
--reinstate NAME, -ri NAME	Reinstate a plugin that was previously removed by providing its name.
--fetch NAME, -f NAME	Get the named plugin description.
--configure NAME, -c NAME	Configure the plugin site properties. Usually requires to restart the associated service to be effective.
--status NAME, -su NAME	Get the status of the service associated to the named plugin.
--start NAME, -sa NAME	Start the service associated to the named plugin.
--stop NAME, -so NAME	Stop the service associated to the named plugin.

Extras

Options	Description

--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

List all the installed plugins:

```
opal plugin --opal https://opal-demo.obiba.org --user administrator
--password password --list
```

Get the plugin description from its name:

```
opal file --opal https://opal-demo.obiba.org --user administrator
--password password --fetch jennite-vcf-store
```

Install a plugin by providing its name and version:

```
opal file --opal https://opal-demo.obiba.org --user administrator
--password password --install jennite-vcf-store:1.0.2
```

Install a plugin by providing its archive file in Opal file system:

```
opal file --opal https://opal-demo.obiba.org --user administrator
--password password --install
/home/administrator/jennite-vcf-store-1.0.2-dist.zip
```

REST API Command

Contents of this page

Synopsis

```
opal rest PATH <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Path

The path on the Opal file system.

Credentials

Authentication is done either by username/password or by public/private key files.

Option	Description

--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	User name to connect to Opal
--password PASSWORD, -p PASSWORD	User's password
--ssl-cert SSL_CERT, -sc SSL_CERT	Path to the certificate (public key) file
--ssl-key SSL_KEY, -sk SSL_KEY	Path to the private key file

Options

Option	Description
ws	Web service path, for instance: /datasource/xxx/table/yyy/variable/vvv
--metod GET|POST|..., -m GET|POST|...	A valid HTTP method
--accept HEADER, -a HEADER	Accept header (default is application/json)
--content-type HEADER, -ct HEADER	Content-Type header (default is application/json)
--force, -f	Skip confirmation

Extras

Options	Description
--help, -h	Displays the command's help message
--verbose, -v	Verbosely executes the command
--json, -j	Output pretty-print JSON

Examples

Get the list of all datasources:

```
opal rest /datasources --opal https://opal-demo.obiba.org --user
administrator --password password --json
```

Get the list of all tables of datasource 'medications':

```
opal rest /datasource/medications/tables --opal https://opal-demo.obiba.org
--user administrator --password password --json
```

Get the list of tables with entity id '6397957' of type 'Participant':

```
opal rest /entity/6397957/type/Participant/tables --opal
https://opal-demo.obiba.org --user administrator --password password --json
```

Opal R and DataSHIELD User Guide

Contents of this Guide

Summary

This guide will describe how to leverage R statistical analysis capabilities over the data stored in one or more Opal servers.

What is R?

R is a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc.

Please consult the R project homepage for further information.

What is DataSHIELD?

DataSHIELD (Wolfson et al., 2010) is a novel method that enables a pooled data analysis to be carried out across several collaborating studies as if one had full access to all of the data from individual participants that might be needed, but, in reality, these data remain completely secure on their host computer at the home base of the study where they were collected or generated. DataSHIELD therefore permits a fully efficient pooled analysis to be undertaken of biomedical data from several studies, even when ethico-legal or other governance restrictions prohibit the release of individual-level data to third parties.

The figure 1 illustrates the basic IT infrastructure that underpins DataSHIELD; it reflects a hypothetical implementation based on a pooled analysis involving data from six studies. The individual-level data that provide the basis of the analysis remain on 'data computers' (DCs) at their home bases. An additional computer is identified as the ‘analysis computer’ (AC). This is the computer on which the primary statistician will type the commands to enact and control the pooled analysis.

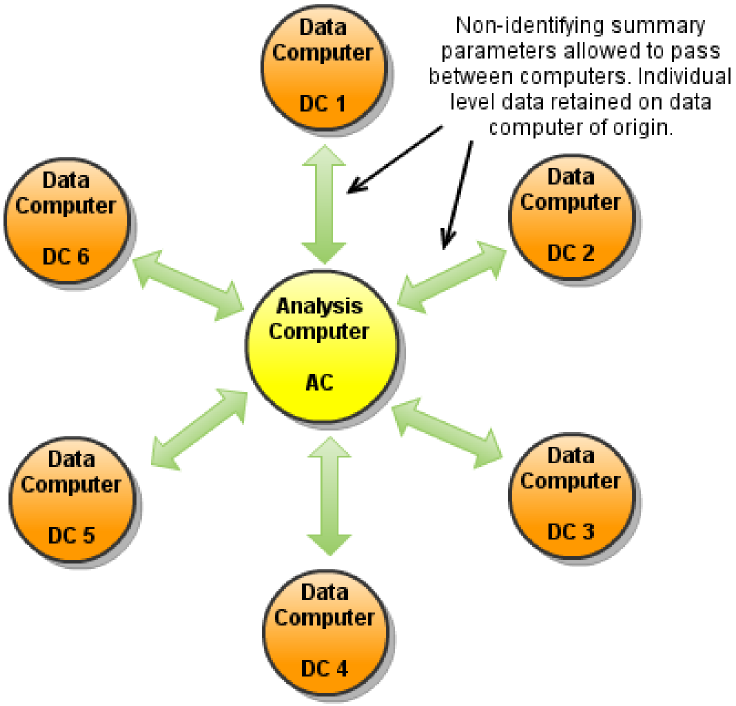

Data Computer DC 1
Non-identifying summary parameters allowed to pass between computers. Individual level data retained on data computer of origin.
Data Computer DC 6
Data Computer DC 2
Analysis Computer AC
Data Computer DC 5
Data Computer DC 3
Data Computer DC 4

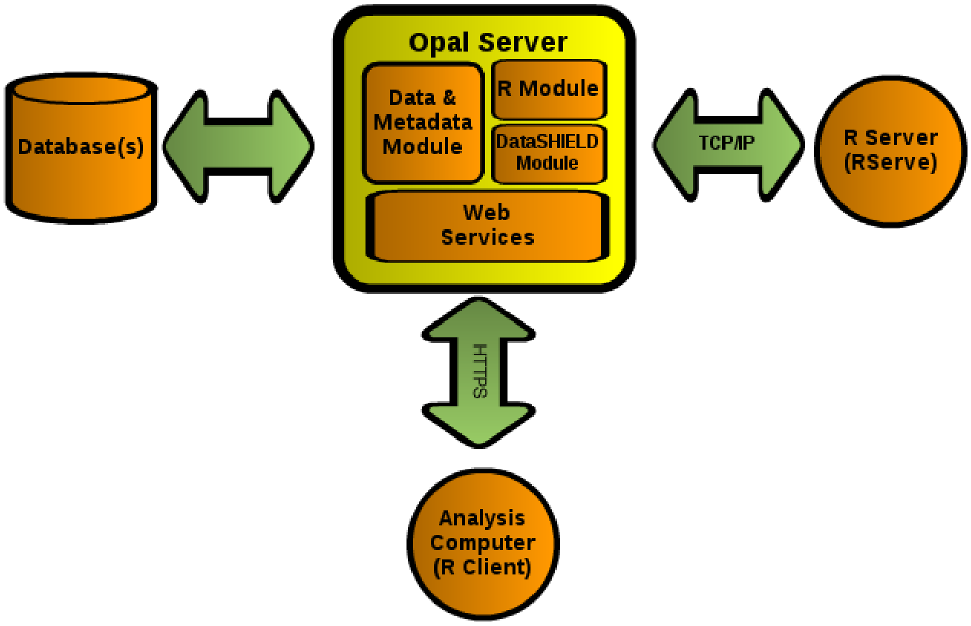
Figure 1: The IT infrastructure underpinning DataSHIELD

Please consult the DataSHIELD project homepage for further information.

R and DataSHIELD implementation in Opal

Opal uses the R statistical environment to implement DataSHIELD. The implementation is made of 3 components:

- an Opal server
- an R server (using Rserve)
- an R package for Opal (installed on the Analysis Computer)



Opal Server Component

This component has several sub components necessary to implement DataSHIELD:

- a data and metadata module
- an R module
- a DataSHIELD module

These sub components are accessible through web services (HTTPs) and interact with each other to provide an extensible and customisable DataSHIELD implementation.

Data and metadata module

Used for obtaining the data necessary for the actual analysis within DataSHIELD. The module also provides metadata that is used for describing the variables involved during the analysis. This metadata provides at least the type of variable (categorical, continuous, logical, etc.), but can also provide higher-level information such as labels, descriptions, etc.

R module

Used for the interaction between an R statistical environment and Opal. Specifically, this module allows pushing data from Opal into an R environment and back. It can also execute arbitrary R code within these environments.

Opal interacts with an R server through Rserve's protocol. This allows the R Server to be on a different machine than the Opal server. It also allows maintaining R separately from Opal.

DataSHIELD module

Built "on top" of the R module, this provides a constrained and customisable access to the R environment. Specifically, this module allows pushing data from Opal into R, but does not allow reading this data unless it has first been "aggregated".

The term "aggregated" here means that the data in R must go through a method that will summarize individual-level data into another form that removes the original individual-level data. For example, obtaining the length of a vector, obtaining the summary statistics of a vector (min, max, mean, etc.)

It is these methods that are customisable. That is, administrators of the Opal server can add, remove, modify and create completely custom "aggregating" methods that are provided to DataSHIELD clients.

Web Services

Interaction between these modules and their clients is done through Web Services.

R Server Component

R is made accessible to Opal through the `Rserve` library. This allows running R commands from several remote clients. Doing so allows running R and Opal on different machines if necessary.

Note that this R Server will eventually contain individual-level data (it will be pushed there by the Opal server). This R server should be secured just like other machines involved in handling individual-level data. This data is not made directly available to Opal clients.

R Clients (Analysis Computers)

The interaction between the analysis computer and Opal is done through another R environment running on the AC. To support these interactions, Opal provides an R package that can be installed using normal R functionalities (CRAN).

Clients can then use this package to authenticate to Opal instances and interact with the DataSHIELD methods offered by these servers.

Opal R User Guide

Contents of this Guide

Summary

This guide will teach you some basics about how to access to Opal variables and data using the R programming language.

Prerequisites

On client side, R is to be available. See the R installation documentation that match your system.

Installation

On Linux

Installing the R packages requires the header files for the `libcurl` system library. To install this on ubuntu/debian (as root):

```
# sudo apt-get install libcurl4-openssl-dev
```

Then you can install the Opal package and its dependencies with this command within an R session:

```
install.packages('opal', repos=c('http://cran.rstudio.com/',
'http://cran.obiba.org'), dependencies=TRUE)
```

On Windows

Installing the Opal R package on Windows requires that the package's dependencies be already installed:

```
install.packages(c('RCurl', 'rjson'), repos=c('http://cran.rstudio.com/',
'http://www.stats.ox.ac.uk/pub/RWin/'))
```

Once these are installed, the opal package can be installed like so:

```
install.packages('opal', repos='http://cran.obiba.org', type='source')
```

Usage

Accessing Opal data using R is straightforward:

```
# load opal library
library(opal)

# get a reference to the opal server
o <- opal.login('administrator','password','https://demo.obiba.org:8443')

# assign some data to a data.frame
opal.assign(o,'HOP','mica_demo.HOP',variables=list('GENDER','PM_BMI_CONTIN
UOUS'))

# do some analysis on the remote R session
opal.execute(o,'summary(HOP)')

# get the remote data.frame on the client
D <- opal.execute(o,'HOP')
head(D)

# clean remote R session
opal.logout(o)
```

For a more complete example see this Opal R test script.

Security

As the user is authenticated against Opal, the authorizations granted to this user applies. If user is only allowed to access to the variables and not the data, the data assignment to R will fail.

In addition to the data and variables related permissions, the user must have been granted the permission to use the R service.

It is highly recommended to access to a Opal server through the secured protocol HTTPS (Opal address starting with `https://`).

Advanced users can login to Opal by providing a key pair: certificate and private key. Example:

```
credentials <- list(
  sslcert='my-publickey.pem',
  sslkey='my-privatekey.pem')

o <- opal.login(url='https://demo.obiba.org:8443', opts=credentials)
```

Opal DataSHIELD User Guide

Contents of this Guide

Summary

This guide will teach you some basics about how to access to Opal variables and data using the R programming language.

Prerequisites

On client side, R is to be available. See the R installation documentation that match your system.

Installation

On Linux

Installing the R packages requires the header files for the `libcurl` system library. To install this on ubuntu/debian (as root):

```
# sudo apt-get install libcurl4-openssl-dev
```

Then you can install the Opal package and its dependencies with this command within an R session:

```
install.packages('opal', repos=c('http://cran.rstudio.com/',
'http://cran.obiba.org'), dependencies=TRUE)
```

On Windows

Installing the Opal R package on Windows requires that the package's dependencies be already installed:

```
install.packages(c('RCurl', 'rjson'), repos=c('http://cran.rstudio.com/',
'http://www.stats.ox.ac.uk/pub/RWin/'))
```

Once these are installed, the opal package can be installed like so:

```
install.packages('opal', repos='http://cran.obiba.org', type='source')
```

Usage

Setting up User Permissions

Using DataSHIELD requires two kind of permissions:

- 'Use' permission to DataSHIELD services: see DataSHIELD Permissions section for more details.
- At least 'View dictionary and summaries' permission to some data descriptions: see Table Permissions to know how to grant access to a table.

These access rights can be granted to a user or a group of users.

Deploying DataSHIELD packages in Opal

Each Opal must be configured the same way so that same computation is done in each Opal for one client request. This is done by relying on DataSHIELD-R packages repository.

See documentation about DataSHIELD Packages Administration. See also DataSHIELD documentation for Administrators.

In the following example, the *dsbase* package is to be installed.

DataSHIELD Usage

First thing required to use DataSHIELD is to load *datashieldclient*, the DataSHIELD base package for the client, into your R environment:

```
# Install datashieldclient and dependecies if not already done
install.packages('datashieldclient', repos=c(getOption('repos'),
'http://cran.obiba.org'), dependencies=TRUE)

# Load datashieldclient library
library(datashieldclient)
```

Create an Opal Object

Every method in the opal package has a required parameter of type 'opal'. This type of object can be obtained by calling the `opal.login` method . This is also th method used to authenticate with an Opal server.

```
# Create a Opal object
o <- opal.login('username', 'password', 'https://opal.domain.org')
```

Alternatively, the `url` parameter can be specified as a `list` to login to multiple Opal instances with the same credentials (username/password) everywhere.

```
# Create a Opal object for each Opal url
opals <- opal.login('username', 'password', list('https://opal.domain.org',
'https://opal.anotherdomain.org'))
```

This method returns an 'opal' (or a list thereof) object that can be passed to other methods later. The return value of this method should be stored in a variable for later use.

Finally, additional options can be specified using the `opts` parameter. This list is passed to the `curlOptions` method for setting HTTP options. Here are some useful ones:

Option	Description

`ssl.verifypeer`	Set to `0` to allow HTTPs connections to servers that provide self-signed certificates.
`ssl.verifyhost`	Set to `0` to allow HTTPs connections to servers that provides a certificate for a different hostname.
`sslversion`	Specify the SSL version number.

Additional information is provided here.

Invoking DataSHIELD Methods

As mentioned previously, all DataSHIELD methods require an argument of class 'opal' (returned by opal.login). This allows working with multiple opal instances in a single client:

```
# Login in each Opal
studyA <- opal.login('username', 'password', 'http://opal.studya.org')
studyB <- opal.login('username', 'password', 'http://opal.studyb.net')

# Invoke some datashield methods
datashield.assign(studyA, ...)
datashield.assign(studyB, ...)
```

Since DataSHIELD is always using multiple Opal instances, the same methods are also able to work on a list of opal objects and will return a list of results.

```
# Login in each Opal
studyA <- opal.login('username', 'password', 'http://opal.studya.org')
studyB <- opal.login('username', 'password', 'http://opal.studyb.net')

opals <- list(StudyA=studyA, StudyB=studyB)

# Invoke a datashield method for all elements of 'opals'
datashield.newSession(opals, ...)
```

This allows transforming for loops into single calls:

```
# Instead of this:
for(k in opals) {
  datashield.assign(k, ...)
}
# Write this:
datashield.assign(opals, ...)
```

Obviously, the downside is that the arguments are the same to all opal instances. If this is not the case, then a manual call to each opal instance will always be required.

Working with Server-Side R

Assignments

Opal can push data into the server-side R environment and assigned to a particular symbol. This is done using the `datashield.assign` method. Opal can push a variable, a table (with all its variables) or even a datasource (with all tables and variables) into R and assign it to a R symbol. Opal data to be pushed are identified by Opal Fully Qualified Names.

```
# Assign the 'opal-data.Table:Variable' to the VAR symbol
datashield.assign(opals, 'VAR', 'opal-data.Table:Variable')

# Assign all variables from 'opal-data.Table' to the TBL symbol
datashield.assign(opals, 'TBL', 'opal-data.Table')

# Assign some enumerated variables from 'opal-data.Table' to the TBL symbol
as a data.frame
datashield.assign(opals, 'TBL', 'opal-data.Table',
variables=list('VAR1','VAR2'))

# Assign all continuous variables from 'opal-data.Table' to the TBL symbol
as a data.frame
datashield.assign(opals, 'TBL', 'opal-data.Table',
variables='nature().any("CONTINOUS")')
```

The `datashield.assign` method can also be used to assign arbitrary R code on the server.

```
# Arbitrary R data can also be assigned on the server.
# This requires the use of the quote() function to protect from local
evaluation.
datashield.assign(opals, 'some.data', quote(c(1:10)))
datashield.assign(opals, 'other.data', quote(my.func(some.data)))
```

The remote R symbols can be listed and deleted.

```
# List the symbols in each Opal for the current datashield session
datashield.symbols(opals)
# Remove a symbol from each Opal for the current datashield session
datashield.rm(opals, 'TBL')
```

Aggregations

As per the DataSHIELD method, only aggregated data may be returned by the server. The server is configured with a set of methods provided to the DataSHIELD clients. The usage pattern is as follows:

1. clients manipulate the server-side R environment (assign data, transform data, etc.)
2. clients request an aggregate of some value in the R environment
3. server extracts the requested value from the R environment
4. server executes the aggregation method on the requested the data in a freshly created environment
5. server returns aggregate data to clients.

This allows a broad range of possibilities to clients, but all "read" operations are controlled by the server and should not permit access to individual-level data.

The aggregation methods are defined by the server and so are configurable: see Aggregation Methods section in Opal Web Application User Guide to know how to manage these methods. But some should always be available since they are required to implement the DataSHIELD methods.

```
# Assign some Opal data in the environment
datashield.assign(opals, 'BMI', 'opal-data.Impedence:BMI')

# Use the 'length' aggregating method to retreive the length of the vector
in each Opal
datashield.length(opals, 'BMI')

# Alternatively, use the 'aggregate' method to invoke 'length'
# This form is used to invoke methods not defined by default
datashield.aggregate(opals, 'length(BMI)')
```

Generalized Linear Model (glm) Example

```
# Login to all Opal instances. The 'ssl.verifypeer' parameter is used to
login to Opal instances that use self-signed certs.
opals<-datashield.login('username', 'password',
list(S1='https://demo.obiba.org:8443',

S2='http://opal.obiba.org',

S3='http://localhost:8080'), list(ssl.verifypeer=0))

# Push the data we want to work with in the server-side R
datashield.assign(opals, 'ds.demo', 'ds-demo.Simulated')

# Optional step: convert the pairlist to a data.frame.  This step
simplifies the call to datashield.glm
datashield.assign(opals, 'ds.frame<-data.frame(as.list(ds.demo))')

# Treat snp as factors (snp.f)
datashield.assign(opals, 'snp.f<-as.factor(ds.frame$snp)')

# Treat smoke as factors (smoke.f)
datashield.assign(opals, 'smoke.f<-as.factor(ds.frame$smoke)')

# Run glm. Note the usage of "quote()" to prevent early evaluation.
datashield.glm(opals, CC~1+study.2+study.3+bmi+bmi.study.3+snp.f*smoke.f,
quote(binomial))
```

Extending DataSHIELD

DataSHIELD is extensible; new aggregating methods can be defined on Opal servers such that any client can make use of them. It is also described here: Aggregation Methods section in Opal Web Application User Guide

DataSHIELD administrators can define two types of aggregating methods: `R Function` or `R Script`.

R Function Aggregating Methods

This type of aggregating method is used to directly invoke an R function on the data from the user's R environment. Because no pre-condition can be defined for these methods, they should be limited to very simple methods such as 'length'. Any R Function method can be written as an R Script method and may allow more control over what is being aggregated.

R Script Aggregating Methods

These types of aggregating methods are free-form R Scripts. They can invoke any R function available and also add pre and post conditions to what is being aggregated. Using this type of method requires more work for administrators, but allow more flexibility in terms of data security.

For example, pre conditions could validate that the input data has a minimum size before invoking a summarizing function on it. Post conditions could remove some unsafe data from the result before passing it back to clients.

Contributing to DataSHIELD Packages

DataSHIELD packages sources are hosted on GitHub.

Some DataSHIELD developers documentation is also available.

Opal Reporting User Guide

Contents of this Guide

Summary

This guide provides information about how to design reports with R over Opal data.

R being a programming language, any text editor could be used. In this guide we recommend to use the RStudio editor as it has reporting features integrated. RStudio is cross-platform and is free of charge.

Download RStudio IDE

Prerequisites

- having R package "opal" installed
- having access to a Opal server

Design of a Report Tutorial

A report in Opal is essentially a R script enhanced with presentation directives. This reporting capability is brought by the knitr R package. As the report IS a R script, it can be executed in different contexts:

- R console
- RStudio editor
- Shell script
- Opal

See more information about Report Execution Flows.

The following steps will walk you through the design of a report, tested in a development environment (R console/RStudio), then deployed in a production environment (Shell script/Opal).

First Step: Write a R Script

Report data are coming first, so start with writing a R script that:

- connects to a Opal server

- assign some Opal data to the remote R session
- analyse, transform the data from the remote R session
- end remote R session

Example

See an example of such a script: opal.R.

Run it in a R console or RStudio.

Second Step: Turn R script to R markdown

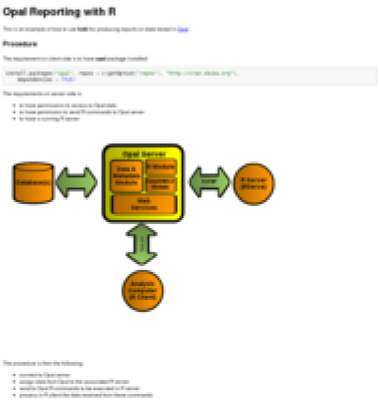
The R script can be enhanced with presentation directives as specified by knitr. We will chose the specific R report format based on Markdown. Detailed documentation can be found in the R Markdown article.

Example

See an example of such a report: opal-dev.Rmd.

You can run it in RStudio as described in the Using R Markdown article.

See output here:



Opal Reporting with R

This is an example of how to use **knitr** for producing reports on data stored in Opal.

Procedure

The requirement on client side is to have **opal** package installed:

```
install.packages("opal", repos = c(getOption("repos"), "http://cran.obiba.org"),
    dependencies = TRUE)
```

The requirements on server side is:

- to have permissions to access to Opal data
- to have permission to send R commands to Opal server
- to have a running R server

The procedure is then the following:

- connect to Opal server
- assign data from Opal to the associated R server
- send to Opal R commands to be executed in R server
- process in R client the data received from these commands

Third Step: Prepare for Deployment

As you might have noticed the opal-dev.Rmd contains the credentials of the user connecting to the Opal server. These can be externalized. Credentials will be provided by the context of execution as R options:

- Shell script
- Opal

See documentation about opal.login function for available R options.

Example

See an example of a production report opal-prod.Rmd. Note that no user credentials is provided. RStudio cannot execute it as usual as the editor does not knit the report in the current R session.

To execute this report you can use the opal.report helper function that will knit it for you. See an example of a R shell script running it: opal-exec.R
.

For executing it in Opal, see instructions on how to specify the R options in the Reports Administration documentation.

Final Step: Schedule Report Execution

Once a report design is done, it is possible to register it in Opal in order to:

- publish it so that it can be executed manually by other users,
- execute it periodically,
- archive and publish the reports generated.

For more details see Reports Administration.

If you have written a shell script, such as opal-exec.R example, it can be executed as any cron task.

Advanced Examples

See advanced examples in the table folder, where opal-table.Rmd features:

- R Markdown sub-reports,
- inline css-styling,
- access to Opal variables description.

The result of this report is a document that presents the data dictionary of a Opal table with figures and summary statistics.

See output here:

Opal Data Dictionary with R

Opal Data Dictionary with R

This is an example of how to use **knitr** for producing a data dictionary on data stored in Opal.

The table **StandingHeight** is about "Participant".

It has 21 variables and 5110 entries.

Variables

- InstrumentRun.user
- InstrumentRun.timeStart
- InstrumentRun.timeEnd
- InstrumentRun.otherContraindication
- InstrumentRun.instrumentBarcode
- InstrumentRun.Contraindication.code
- InstrumentRun.Contraindication.type
- RES_THIRD_HEIGHT
- CALC_ABS_DIFF_FIRST_AND_SECOND_HEIGHT.captureMethod
- CALC_ABS_DIFF_FIRST_AND_THIRD_HEIGHT
- RES_SEC_HEIGHT
- RES_FIRST_HEIGHT
- RES_FIRST_HEIGHT.captureMethod
- CALC_ABS_DIFF_FIRST_AND_THIRD_HEIGHT.captureMethod
- RES_SEC_HEIGHT.captureMethod
- CALC_AVG_HEIGHT.captureMethod
- CALC_ABS_DIFF_SECOND_AND_THIRD_HEIGHT
- CALC_ABS_DIFF_SECOND_AND_THIRD_HEIGHT.captureMethod
- CALC_ABS_DIFF_FIRST_AND_SECOND_HEIGHT
- CALC_AVG_HEIGHT
- RES_THIRD_HEIGHT.captureMethod

InstrumentRun.user

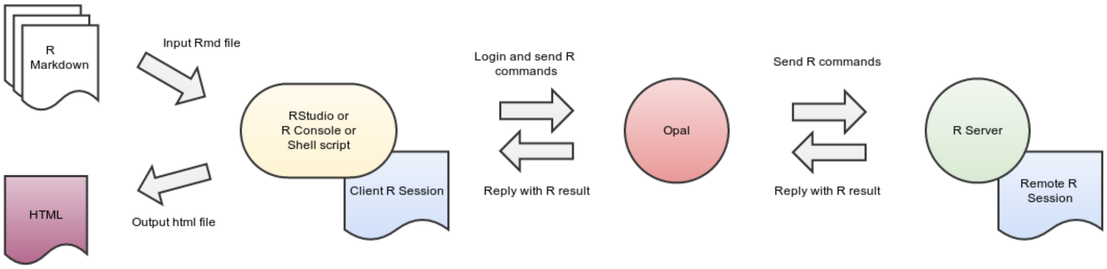
Properties

Property	Value
Entity type	Participant
Value type	text
Unit	
Mime Type	

Report Execution Flows

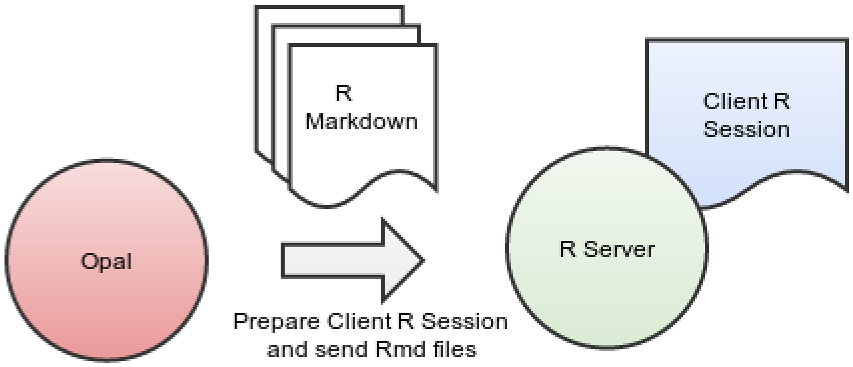
Local R Session

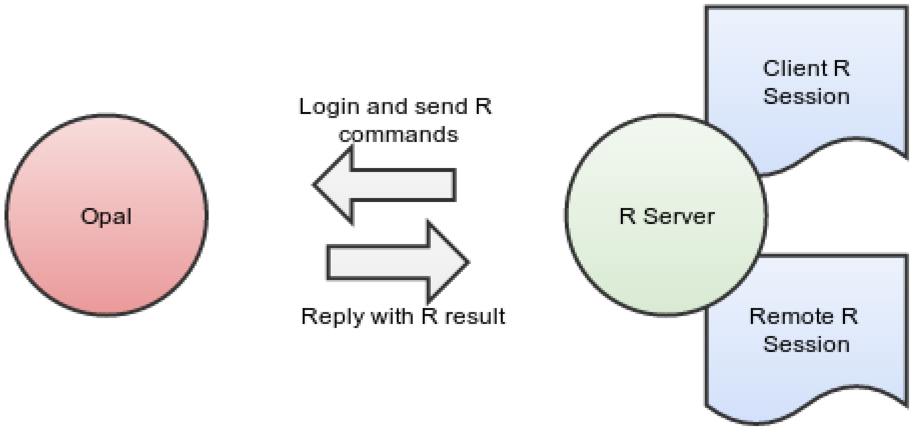
When executed in the context of RStudio, R console or Shell script, the communication flow is:

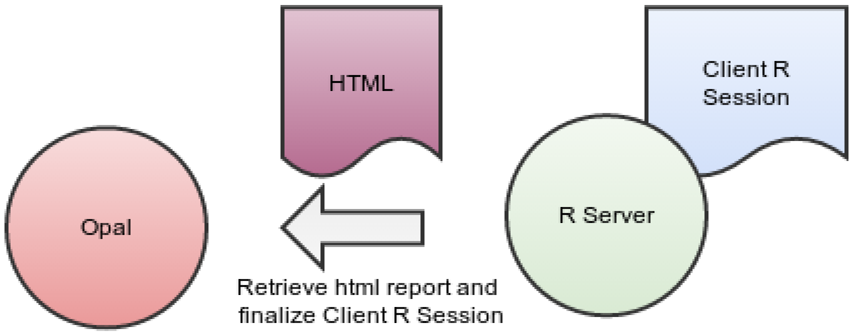


Opal R Session

When executed by Opal the communication flow is:







Magma Javascript API

Contents of this Guide

Introduction

The Magma Javascript API is there to allow accession of Magma variable catalogue. It is used for Onyx and Opal configuration.

Syntax

Selectors and Execution context

The API offers a simple way to access to variables and variable value for a value set, based on Magma naming schema. A javascript is always defined within a context that defines implicitly the object to which the selection is applied.

See selectors description: $ and $var.

Chaining

Methods from this API return an object to which a method can be directly applied. This allow method calls chaining.

Examples

```
$('DO_YOU_SMOKE').any('DNK', 'PNA').not()
```

Methods

Global Methods

Method	Description
$, $val, $value	Return the variable value within the current value set.
$this	Return the variable value within the current view value set.
$group	Return a map of variable values in the same group of occurrence within the current value set.
$id, $identifier	Return the entity identifier within the current value set.
$join	Return the joined variable value referenced by a variable value within the current value set.
$var, $variable	Returns the variable object at the given name.
log	Provides 'info' level logging of messages and variable values.
now	Returns the current date time wrapped in a value object.

newValue	Creates a new value.
newSequence	Creates a new value sequence.
$created	Return the value set creation time.
$lastupdate	Return the value set last update time.
source	Load a javascript file.

Variable Methods

These methods are to be used when execution context is a Variable. *CATEGORICAL*

Method	Description
attribute	Get the variable attribute value with the given name.
name	Get the name of the variable as a Text value.
type	Returns the value type of the variable, as a Text value.
entityType	Returns the entity type of the variable, as a Text value.
refEntityType	Returns the referenced entity type of the variable, as a Text value.
repeatable	Returns if the variable is repeatable, as a Boolean value.
occurrenceGroup	Returns the occurrence group of the variable, as a Text value.
mimeType	Returns the mime type of the variable, as a Text value.
unit	Returns the unit of the variable, as a Text value.
nature	Returns the nature of the variable (*CATEGORICAL*, *CONTINOUS*, etc.), as a Text value.
isNumeric	Returns whether the variable value type is *integer* or *decimal*, as a Boolean value.
isDateTime	Returns whether the variable value type is *date* or *datetime*, as a Boolean value.
isGeo	Returns whether the variable value type is *point*, *linestring* or *polygon*, as a Boolean value.

Value Methods

These methods are to be used when execution context is a Value.

Method	Description
all	Returns true when the value contains all specified parameters, false otherwise.
any	Returns true value when the value is equal to any of the parameter, false otherwise.
empty	Returns true value if is operating on a sequence that contains zero values.
isNull	Returns true if the value is `null`.

whenNull	Returns its argument if the value is `null`. This method may allow avoiding an `if/else` block.
map	Uses a lookup table to map the a value to another (which may be computed or derived).
not	Returns the contrary of a boolean value or return if it does not match any of the arguments.
type	Returns or changes the value's type.
value	Returns the javascript value from the `value object`.
length	Returns the length of the value.

Value Sequence Methods

Method	Description
any	Returns true value if of the provided values can be found in the value sequence.
empty	Returns true value if is operating on a sequence that contains zero values.
first	Returns the first value in a value sequence.
firstNotNull	Returns the first not null value in a value sequence.
indexOf	Returns the first position of a value in a value sequence.
last	Returns the last value in a value sequence.
lastIndexOf	Returns the last position of a value in a value sequence.
valueAt	Returns the a value at a specified index within the sequence (0-based).
size	Returns the number of values within a sequence.
map	Map each value in the sequence to another value.
reduce	Returns the reduction of the values within a sequence.
filter	Returns a sequence which values have been filtered using custom javascript predicating function.
subset	Returns a subset of a sequence according to provided begin and end positions.
trimmer	Returns a sequence without null values.
sort	Sorts a sequence in natural order of its values or using a custom javascript comparing function.
max	Returns the maximum value of a value sequence.
min	Returns the minimum value of a value sequence.
avg	Returns the average of a value sequence.
sum	Returns the sum of a value sequence.
stddev	Returns the standard deviation of a value sequence.
push	Adds one or more values after a value to produce a value sequence (deprecated in favor of append).
append	Adds one or more values after a value to produce a value sequence.

prepend	Adds one or more values before a value to produce a value sequence.
insertAt	Inserts one or more values at a given position to produce a value sequence.
join	Joins the text representation of the values in the sequence.
zip	Returns a sequence of values, where each value is the transformation of a tuple of values, the i-th tuple contains the i-th element from each of the argument sequences.
asSequence	Turns a value object into a value sequence object.
isSequence	Returns whether the value is a value sequence object.

Boolean Value Methods

Method	Description
and	Applies the ternary and logic on values.
compare	Returns 0 if the value represents the same boolean value as the argument; a positive integer if the value represents `true` and the argument represents `false`; and a negative integer if this value represents `false` and the argument represents `true`.
eq	Returns if left operand value is equal to right operand value.
not	Returns the contrary of a boolean value or return if it does not match any of the arguments.
or	Applies the ternary or logic on values.

Numeric Value Methods

For Numeric operations a value of integer type is returned if both operands are of the integer type. The decimal type is returned otherwise.

Method	Description
compare	Returns a negative integer, zero, or a positive integer as the value is less than, equal to, or greater than the value argument.
eq	Returns if left operand value is equal to right operand value.
ge	Returns if left operand value is greater equal than right operand value.
gt	Returns if left operand value is greater than right operand value.
le	Returns if left operand value is lower equal than right operand value.
lt	Returns if left operand value is lower than right operand value.
plus	Returns result of first operand value plus second operand value.
minus	Returns result of first operand value minus second operand value.
multiply	Returns result of first operand value multiply second operand value.

div	Returns result of first operand value divided by second operand value.
ln	Returns the natural logarithm (base e) of the value.
log	Returns the base-10 logarithm of the value.
abs	Returns the absolute value.
pow	Returns the value raised to the power of the operand.
sqroot	Returns square root of the value.
cbroot	Returns cubic root of the value.
root	Returns arbitrary root of the value.
round	Returns the rounded value.
group	Group values in a ranges.

Measurement Unit Methods

Methods	Description
unit	Sets the measurement unit of the current value to the specified unit. Returns the current unit when no argument is supplied.
toUnit	Measurement unit conversion: converts the current value into a different measurement unit.

Text Value Methods

Method	Description
compare	Returns a negative integer, zero, or a positive integer as the text value is less than, equal to, or greater than the text value argument.
compareNoCase	Returns a negative integer, zero, or a positive integer as the text value is less than, equal to, or greater than the text value argument ignoring case.
concat	Returns the text type result of first operand concat second operand.
date	Returns a value of date type given a date format pattern.
datetime	Returns a value of datetime type given a date time format pattern.
eq	Returns if left operand value is equal to right operand value.
matches	Used to match a regular expression against a string.
replace	Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.
trim	Returns a copy of the string, with leading and trailing whitespace omitted.

lowerCase	Returns a copy of the string, lower case.
upperCase	Returns a copy of the string, upper case.
capitalize	Returns a copy of the string, with first character of each word capitalized.

Date and Date Time Value Methods

Method	Description
add	Adds days to a value of date time type.
after	Returns true if the date value is after the specified date value(s).
before	Returns true if the date value is before the specified date value(s).
dayOfMonth	Returns the day of month from a date as an integer starting from 1.
dayOfWeek	Returns the day of week from a date as an integer starting from 1 (Sunday).
dayOfYear	Returns the day of year from a date as an integer starting from 1.
format	Returns the text representation of the date formatted by the provided pattern.
hour	Returns the hour of the day for the 12-hour clock (0 - 11).
hourOfDay	Returns the hour of the day for the 24-hour clock.
minute	Returns the minute within the hour.
millisecond	Returns the millisecond within the second.
month	Returns the month of a date as an integer starting from 0 (January).
quarter	Returns the quarter of a date as an integer starting from 0 (Q1).
second	Returns the second within the minute.
semester	Returns the semester of a date as an integer starting from 0 (S1).
time	Returns the number of milliseconds since January 1, 1970, 00:00:00 GMT (epoch time).
weekday	Returns a boolean value indicating whether the date denotes a weekday (between Monday and Friday inclusively).
weekend	Returns a boolean value indicating whether the date denotes a weekend (either Sunday or Saturday).
weekOfMonth	Returns the week of month from a date as an integer starting from 1.
weekOfYear	Returns the week of year from a date as an integer starting from 1.

year	Returns the year value.

Geo Value Methods

Method	Description
longitude	Get the longitude of a *point* value.
latitude	Get the latitude of a *point* value.

Extensions

The following methods only apply when executed by Onyx.

onyx

Allows access to the onyx configuration variables as defined in the `onyx-config.properties` file and to the `lastExportDate`.

Examples

Each the of the following statements return the `Value` for the named Onyx property.

```
onyx('org.obiba.onyx.lastExportDate')
onyx('org.obiba.onyx.participant.purge')
onyx('org.obiba.onyx.webapp.configurationType')
```

Using Selection Statements

To use JavaScript selection statements such as `if-else` and `switch` first convert Magma `ScriptableValues` to native JavaScript values using the `.value()` method. Here are some examples:

if-else

```
if($('BooleanType.blood.contraindicated').value()) {
   log('Blood collection has been contraindicated');
}
```

```
if($('IntegerType.tubes.collected').gt($('IntegerType.tubes.expected')).va
lue()) {
   log('More tubes than expected.');
} else if
($('IntegerType.tubes.collected').lt($('IntegerType.tubes.expected')).valu
e()) {
   log('Less tubes than expected.');
} else {
   log('Collected tubes matches expected tubes.');
}
```

switch

```
switch($('Admin.Participant.gender').value()) {
   case "MALE":
      log('Participant is male.');
   case "FEMALE":
      log('Participant is female.');
   default:
      log('Participant gender is unknown.');
}
```

Advanced Configuration

The javascript engine allows several levels of optimization, usually a compromise between compilation time and execution time.

The optimization level can be specified as a JVM system property, i.e. with the command line argument `-Drhino.opt.level=<level>` where level is a number between -1 (js code is interpreted) and 9 (js code is compiled and optimized as much as possible). See more about Rhino Optimization. In some rare cases the compilation can fail because the script is (very) large: in this situation the optimization level should be set to `-1`.

Global Methods

- joined value selector method
- log method
- now method
- value selector method
- variable selector method
- identifier method
- group selector method
- newValue method
- newSequence method
- this view value selector method
- created method
- lastupdate method
- source method

joined value selector method

$join

Allows joining a variable value to another variable value that provides a entity identifier. The current object is a value set. `$join` will access to a variable value within this value set.

Method of: Global

Syntax

```
$join(name,idname[,flat])
```

Parameters

name is the name of the variable from which the value shall be retrieved.

idname is the name of the variable from which the entity identifier shall be retrieved.

flat specifies that if in case of the join operation result in a value sequence tree, the result should be flatten in a sequence of unique values. Default is *false* and a value sequence tree will be transformed into a sequence of comma separated stringified values.

Examples

Returns the BRAND_NAME of a medication which is identified by the MEDICATION_ID.

```
$join('medications.Drugs:BRAND_NAME','MEDICATION_ID')
```

Given the following datasets:

- *table* has a repeatable variable named *code*

ID	code
aa	Acode1,Acode2
bb	Acode1
cc	Acode3
dd	

- *code_mapper* has a repeatable variable *parent*

ID	parent
Acode1	Bcode1,Bcode2
Acode2	Bcode1,Bcode3
Acode3	

The following script:

```
$join('test.code_mapper:parent','code', true)
```

will return the following value sequencies:

D	flat
aa	Bcode1,Bcode2,Bcode3
bb	Bcode1,Bcode2
cc	
dd	

Without the *flat* option, the following script:

```
$join('test.code_mapper:parent','code')
```

would return the following value sequencies where value sequencies of second order have been stringified:

D	non-flat
aa	"Bcode1,Bcode2","Bcode1,Bcode3"
bb	"Bcode1,Bcode2"
cc	
dd	

See Also

$

log method

log

Provides 'info' level logging of messages and variable values.

Method of: Global

Syntax

```
log(text[, value_i[, ...]])
```

Parameters

- *text* is the text to be logged. May contain place holders for values
- *value_i* is one of the value to be replaced in the *text* by order of appearance.

Examples

```
log('My message');
log('Do you smoke ? {}', $('DO_YOU_SMOKE'));
```

now method

now

Returns the current date time wrapped in a value object.

Method of: Global

Syntax

```
now()
```

Parameters

No parameters.

Examples

Get the current date time.

```
now()
```

value selector method

$, $val, $value

The current object is a value set. $ will access to a variable value within this value set.

Note that when joining several tables in a view and when the named variable is present in more than one of these tables, the value returned is a value sequence. Values of this sequence appear in the order of the tables defined in the view. For more details see also the presentation about m ultilines support. If the values of the sequence are known to be identical (because the same data is repeated in several tables), it is possible to use the firstNotNull method to reduce the sequence to one value.

Note also that if a script returns a value sequence and if the derived variable is not "repeatable", the value sequence is automatically reduced with the firstNotNull strategy.

Method of: Global

Syntax

```
$(name)
// alternate syntax
$val(name)
$value(name)
```

Parameters

name is the name of the variable from which the value shall be retrieved.

If the name is fully qualified, ie. "<datasource>.<table>:<variable>", the lookup will be first done in view's reference tables and if not found will be looked up out of the view's scope.

Examples

Returns the value for current value set and the named variable `DO_YOU_SMOKE`.

```
$('DO_YOU_SMOKE')
```

Returns the value for the current entity from the fully qualified table.

```
$('project.table:SMOKING')
```

> When using a fully qualifying the variable, if the variable is not a variable from the tables referred by the view, the performance could be very poor as it would result in one request on an individual value in the database.

See Also

$group, $id, $join, $this, $var

variable selector method

$var, $variable

Returns the variable object at the given name. The name resolution is done given an execution context.

Method of: Global

Syntax

```
$var(name)
// alternate syntax
$variable(name)
```

Parameters

name is the name that identifies the variable.

Examples

Get the variable with name `DO_YOU_SMOKE`.

```
$var('DO_YOU_SMOKE')
```

See Also

$, $id

identifier method

$id, $identifier

The current object is a value set. `$id` will access to the entity Opal identifier within this value set.

Method of: Global

Syntax

```
$id()
// alternate syntax
$identifier()
```

Parameters

No parameters.

Examples

Correct a particular value of an entity given its identifier. Other entities will return the recorded value for variable `DO_YOU_SMOKE`.

```
$id().map({'778834' : 'NO'}, $('DO_YOU_SMOKE'))
```

See Also

$, $join, $var

group selector method

$group

The current object is a value set. `$group` will access to variable values within this value set. This method will map the values of the variables in the same occurrence group. The group of values considered in the value sequence is the first group matching the provided criteria.

Method of: Global

Syntax

```
$group(name,criteria[,select])
```

Parameters

name is the name of the variable on which the selection criteria is to be applied.
criteria is a value to be compared to or a function for evaluating the matching criteria.

select is the name of the variable from which the value shall be retrieved. This parameter is optional. If not provided a mapping of values for each variables of the same occurrence group is returned (affects script execution performance if useless data are extracted).

Examples

In the following example `StageName`, `StageDuration`, `StageOperator` are repeatable variables in the same occurrence group.
For a value set, the corresponding value sequences could be as follow:

StageName	StageDuration	StageOperator
Consent	123	roger
Questionnaire	234	michelle
Weight	23	jack

`$group` allows to extract a value of a variable given a matching criteria on a variable value in the same occurrence group:

```
// Returns 234
$group('StageName','Questionnaire', 'StageDuration')

// Returns 'michelle'
$group('StageName','Questionnaire','StageOperator')

// Returns 234 as well but extracts also values for the 'StageOperator'
variable
$group('StageName','Questionnaire')['StageDuration']

// Criteria can also be expressed using a function.
// Returns 'jack'
$group('StageDuration', function(value) {
        return value.le(100);
      },'StageOperator')
```

$group handles value sequences. If multiple occurrences match the criteria, the value that is returned is a value sequence.

StageName	ActionType	ActionComment
Consent	START	
Consent	COMPLETE	
Questionnaire	START	
Questionnaire	INTERRUPT	Participant needs a rest
Questionnaire	RESUME	Participant still looks tired
Questionnaire	COMPLETE	Answers cannot be trusted

```
// Returns the value sequence: '','Participant needs a rest','Participant
still looks tired','Answers cannot be trusted'
$group('StageName','Questionnaire','ActionComment')

// Returns the value: 'Participant needs a rest'
$group('ActionType','INTERRUPT','ActionComment')

// Use asSequence() to ensure consistency with values returned for other
participants (if multiple actions are of type 'INTERRUPT' in this example).
// Returns a value sequence with one item: 'Participant needs a rest'
$group('ActionType','INTERRUPT','ActionComment').asSequence()
```

See Also

$, $id, $join, $var

newValue method

newValue

Creates a new value object.

Method of: Global

Syntax

```
newValue(data[,type])
```

Parameters

data is the data to be wrapped in the value object.
type is the value type to be used to interpret the data. If not provided the type is guessed (as much as possible) from the data. See Value Types section for a complete list of types and supported value formats.

Examples

Create some values:

```
// Creates a value of type 'text'
newValue('lorem ipsum')

// Creates a value of type 'integer'
newValue(123)

// Creates a value of type 'integer'
newValue('123','integer')
```

The created value object can be turned into a value sequence:

```
// Creates a value sequence: 123, 234
newValue(123).push(234)

// Creates a value sequence of one item
newValue(123).asSequence()
```

When creating a date/datetime value, the type must be explicit and the data must be textual. See Value Types section for the date/datetime supported formats.

```
// Creates a date value by parsing the provided text using "yyyy-MM-dd"
format
newValue('2013-03-24', 'date')

// Creates a datetime value by parsing the provided text using "yyyy-MM-dd
HH:mm" format
newValue('2013-03-24 10:56', 'datetime')

// Currently not supported
newValue(new Date())
```

See Also

asSequence, newSequence

newSequence method

newSequence

Creates a new value sequence object. A value sequence *is-a* value object but also *has-some* value objects (see Value Sequence Methods).

Method of: Global

Syntax

```
newSequence(data[,type])
```

Parameters

data is the data to be wrapped in the value object. If an array is provided, each item will be wrapped in a value object and added to the value sequence.
type is the value type to be used to interpret the data. If not provided the type is guessed (as much as possible) from the data. See Value Types section for a complete list of types and supported value formats.

Examples

Create some value sequences:

```
// Creates a value sequence of type 'text' with one item
newSequence('lorem ipsum')

// Creates a value sequence of type 'integer' with one item
newSequence(123)

// Creates a value sequence of type 'integer' with one item
newSequence('123','integer')

// Creates a value sequence of type 'text' with 3 items
newSequence(['a','b', 'c'])
```

See Also

asSequence, newValue

this view value selector method

$this

The current object is a value set in a view. `$this` will access to a variable value within this value set.

Method of: Global

Syntax

```
$this(name)
```

Parameters

name is the name of the variable in the current view from which the value shall be retrieved.

Examples

Returns the value for current value set and the named variable `DERIVED_VAR`.

```
# Get the value of the DERIVED_VAR, which is a variable defined in the view
$this('DERIVED_VAR')

# $this is equivalent but much faster than the explicit variable definition
$('my_datasource.my_view:DERIVED_VAR')
```

See Also

$, $group, $id, $join, $var

created method

$created

The current object is a value set. `$created` will access to the creation timestamp within this value set. The creation timestamp is the date time when the values for a given participant were imported into opal.

Method of: Global

Syntax

```
$created()
```

Parameters

No parameters.

Examples

Check if a value set was created after a given date time.

```
$created().after(newValue('2014-05-05 10:30', 'datetime'))
```

See Also

$lastupdate

lastupdate method

$lastupdate

The current object is a value set. `$lastupdate` will access to the last update timestamp within this value set. The last update timestamp is the date time when the values for a given participant were updated in opal. It usually close to the creation date time unless the values were overridden.

Method of: Global

Syntax

```
$lastupdate()
```

Parameters

No parameters.

Examples

Check if a value set was created after a given date time.

```
# compare to a date
$lastupdate().after(newValue('2014-05-05', 'date'))

# compare to a datetime (ISO 8601 format)
$lastupdate().after(newValue('2015-02-18T11:56:27.280-0500', 'datetime'))
```

See Also

$created

source method

source

This method allows to load a javascript library by specifying a path to a file. The result of this method is like in-lining javascript code in the current script. The loaded javascript libraries can be used for defining frequently used functions and easily reuse them across multiple scripts.

Method of: Global

Syntax

```
source(path)
```

Parameters

path is the path to the javascript file. The path must be absolute in the opal file system.

Examples

Define a javascript file with the following code:

/project/questionnaires/lib/age.js

```
/*
 * Calculate the age from 2 arguments:
 *   birthDate: date of birth value
 *   otherDate: date to compare with
 */
function age(birthDate, otherDate) {
    var years = otherDate.year().minus(birthDate.year());

    if (otherDate.month().lt(birthDate.month()).value() ||
        otherDate.month().eq(birthDate.month()).value() &&
otherDate.dayOfMonth().lt(birthDate.dayOfMonth()).value()) {
        years = years.minus(1);
    }

    return years;
}
```

Then load this javascript file by specifying its location:

```
// load the library that defines the age() function
source('/project/questionnaires/lib/age.js');

// then use the age() function:
//   age at the time of the interview
age($('DATE_OF_BIRTH'), $('INTERVIEW_DATE'));
//   or age at the time of the script execution
age($('DATE_OF_BIRTH'), now());
```

See Also

$

Variable Methods

- attribute method
- name method
- repeatable method

- variable type method
- variable entity type method
- variable mime type method
- variable occurrence group method
- variable referenced entity type method
- variable unit method

attribute method

attribute

Get the variable attribute value with the given name.

Method of: Variable

Syntax

```
attribute(name)
```

Parameters

`name` is the name that identifies the attribute.

Examples

Get the value corresponding to the 'stage' attribute:

```
$var('AVG_SITTING_HEIGHT').attribute('stage')
```

See Also

entityType, mimeType, name, occurrenceGroup, refEntityType, repeatable, type, unit

name method

name

Get the name of the variable as a Text value.

Method of: Variable

Syntax

```
name()
```

Parameters

No parameters.

Examples

If the scriptable context is a variable, get its name.

```
name()
```

See Also

attribute, entityType, mimeType, occurrenceGroup,refEntityType, repeatable, type, unit

repeatable method

repeatable

Returns if the variable is repeatable, as a Boolean value.

Method of: Variable

Syntax

```
repeatable()
```

Parameters

No parameters.

Examples

```
$var('DO_YOU_SMOKE').repeatable()
```

See Also

attribute, entityType, mimeType, name, occurrenceGroup, refEntityType, type, unit

variable type method

type

Returns the value type of the variable, as a Text value.

Method of: Variable

Syntax

```
type()
```

Parameters

No parameters.

Examples

```
$var('DO_YOU_SMOKE').type()
```

See Also

attribute, entityType, mimeType, name, occurrenceGroup, refEntityType, repeatable, unit

variable entity type method

entityType

Returns the entity type of the variable, as a Text value.

Method of: Variable

Syntax

```
entityType()
```

Parameters

No parameters.

Examples

```
$var('DO_YOU_SMOKE').entityType()
```

See Also

attribute, mimeType, name, occurrenceGroup, repeatable, type, unit

variable mime type method

mimeType

Returns the mime type of the variable, as a Text value.

Method of: Variable

Syntax

```
mimeType()
```

Parameters

No parameters.

Examples

```
$var('DO_YOU_SMOKE').mimeType()
```

See Also

attribute, entityType, name, occurrenceGroup, repeatable, refEntityType, type, unit

variable occurrence group method

occurrenceGroup

Returns the occurrence group of the variable, as a Text value. The occurrenceGroup is a way to group repeatable variables of a table.

Method of: Variable

Syntax

```
occurrenceGroup()
```

Parameters

No parameters.

Examples

```
$var('DO_YOU_SMOKE').occurrenceGroup()
```

See Also

attribute, entityType, mimeType, name, repeatable, refEntityType, type, unit

variable referenced entity type method

refEntityType

Returns the referenced entity type of the variable, as a Text value. When the referenced entity type is not null, the values of the variable are identifiers of entities with the referenced type.

Method of: Variable

Syntax

```
refEntityType()
```

Parameters

No parameters.

Examples

```
// 'Participant' is taking a medication identified by a 'Drug'
identification number
$var('MEDICATION_DIN_1').refEntityType()
```

See Also

attribute, entityType, mimeType, name, occurrenceGroup, repeatable, type, unit

variable unit method

unit

Returns the unit of the variable, as a Text value.

Method of: Variable

Syntax

```
unit()
```

Parameters

No parameters.

Examples

```
$var('WEIGHT').unit()
```

See Also

attribute, entityType, mimeType, name, occurrenceGroup, refEntityType, repeatable, type

Value Methods

- all method
- any method
- empty method
- isNull method
- whenNull method
- map method
- not method
- value method
- value type method
- length method

all method

all

Returns true when the value contains all specified parameters, false otherwise. Note that this method will always return false if the value is null.

Method of: Value

Syntax

```
all(param_1[, param_i[, ...]])
```

Parameters

param_i a string or a `Value` to be compared to.

Examples

```
$('CategoricalVar').all('CAT1', 'CAT2')
```

See Also

any, empty, isNull

any method

any

Returns true value when the value is equal to any of the parameter, false otherwise. Note that this method will always return false value if the value is null. When applied to a value sequence, it is applied to each of its values and then returns true if at least one of the value verifies the comparison.

Method of: Value, Value Sequence

Syntax

```
any(param_1[, param_i[, ...]])
```

Parameters

param_i a string or a `Value` to be compared to or a comparison function that takes value argument (makes sense when applied to value sequences).

Examples

```
$('CategoricalVar').any('CAT1', 'CAT2')
```

Usage of a comparison function with the example of a (repeatable) variable representing a sequence of measures:

```
// check if there is any value greater than 100 in a value sequence
$('Measures').any(function(value, index) { return value.le(100) })

// equivalent to
$('Measures').filter(function(value, index) { return value.le(100)
}).empty().not()
```

See Also

all, empty, isNull, not

empty method

empty

Returns true value if is operating on a sequence that contains zero values. Otherwise false value is returned.

Method of: Value, Value Sequence

Syntax

```
empty()
```

Parameters

No parameters.

Examples

```
$('Admin.Interview.exportLog.destination').empty()
```

See Also

all, any, isNull, size

isNull method

isNull

Returns true if the value is null.

Method of: Value

Syntax

```
isNull()
```

Parameters

No parameters.

Examples

```
$('MyVar').isNull()
```

See Also

all, any, empty

whenNull method

whenNull

Returns its argument when the value is `null`. Using this method may avoid the use of an `if/else` block. It can be used to ensure that a method chain never returns a `null` value.

Method of: Value

Syntax

```
whenNull(newValue)
```

Parameters

`newValue` the value to return when the original value is `null`.

Examples

```
// the call to any() may result is null. Thus, adding whenNull ensures that
the chain never returns null.
$('MyVar').any('YES').whenNull(false);

// This complex chain, may result in null for many reasons. Again, adding
the whenNull ensures the result is never null.
$('MyVar').or(
  $('SomeOtherVar').and($('YetAnotherVar'))
).whenNull(false);

// Returns a text Value when null
$('MyTextVar').whenNull('foo');
$('MyTextVar').whenNull($('AnotherTextVar'));

// Returns a integer Value when null
$('MyIntegerVar').whenNull(999);
$('MyIntegerVar').whenNull('999');

// Applies to each value in a value sequence
$('MyRepeatableVar').whenNull(99);
```

See Also

isNull

map method

map

Uses a lookup table to map the a value to another (which may be computed or derived). When the value to be mapped is not found in the association table, then:

- if a value is specified for `null` values, this value is returned,
- else if a default value is specified, the default value is returned,
- else `null` is returned.

Another way to use this method is to provide a mapping javascript function, especially useful for value sequences.

Method of: Value, Value Sequence

Syntax

```
map({key_1:value_1[, key_i:value_i[, ...]]}[, defaultValue[, nullValue]])
map(mapper)
```

Parameters

A list of key/value pairs:

- `key_i` the value to be mapped.
- `value_i` the mapping value, explicitly provided or the result of an operation or a function.

defaultValue (optional) value to return when the lookup value is not found in the list of key/value pairs. (since Opal 1.5.1 and Onyx 1.8.3)

nullValue (optional) value to return when the lookup value is null.

mapper a mapping javascript function called for each value (with arguments: value and index of the value in the sequence) and that returns the resulting mapped value.

Examples

Simple constant lookup table

Value	Mapping Value
'NO'	0
'YES'	1
'DNK'	8888
anything else	9999

```
$('SMOKE').map(
  {'NO':0,
   'YES':1,
   'DNK':8888}, 9999)
```

Lookup table with computed mapped values

Value	Mapped Value
'AGE'	the value of the 'SMOKE_ONSET_AGE' variable
'YEAR'	compute the value 'SMOKE_ONSET_YEAR' minus the year part of the 'BIRTH_DATE' variable
'DNK'	8888
'PNA'	9999
null	7777

```
$('SMOKE_ONSET').map(
  {'AGE':$('SMOKE_ONSET_AGE'),
   'YEAR':$('SMOKE_ONSET_YEAR').minus($('BIRTH_DATE').year()),
   'DNK':8888,
   'PNA':9999}, null, 7777)
```

> Note that all the mapped values will be computed, regardless of the input. If the computation is expensive, consider using a function to compute it, as it will only be invoked when required. See below for an example.

Accepts sequences and returns sequences. In the following example, if the input is 'FRENCH,ENGLISH', the output will be '0,1'.

```
$('LANGUAGES_SPOKEN').map({'FRENCH':0, 'ENGLISH':1});
```

The value can also be computed by executing an arbitrary function. This can be used to lazily evaluate mapped values.

```
// Can execute function to calculate lookup value
$('BMI_DIAG').map(
  {'OVERW': function(value) {
              // 'OVERW' is passed in as the method's parameter
              // some expensive computation happens only when the input
actually is 'OVERW'
              return expensiveValue;
            },
   'NORMW': 0
  });
```

Functions can also be passed to define the default value and/or the null value.

```
$('LANGUAGES_SPOKEN').map({'FRENCH': 'FR', 'ENGLISH': 'EN'}, function(val)
{ return val.substring(0, 2) }, function() { return '??' })
```

A mapping function can pe provided in place of the mapping object and the additional values (null and default).

```
$('LANGUAGES_SPOKEN').map(function(val,idx) { return val.isNull().value() ?
'??' : val.value().substring(0,2) })
```

See Also

type, value, filter, reduce

not method

not

Returns the contrary of a boolean value or return if it does not match any of the arguments.

Method of: Value, Boolean Value

Syntax

```
not([value1[,value2[, ...]]])
```

Parameters

value_i (optional) a `Value` or a string to be compared to.

Examples

Get the contrary of a Boolean value:

```
$('BooleanVar').not()
```

```
$('CategoricalVar').any('CAT1').not()
```

Check if a value is not any of the specified strings:

```
$('CategoricalVar').not('CAT1', 'CAT2')
```

Check if a value is not any of the specified values:

```
$('CategoricalVar').not($('OtherCategoricalVar'))
```

See Also

and, any, isNull

value method

value

Returns the javascript value from the `Value`.

Method of: Value

Syntax

```
value()
```

Parameters

No parameters.

Examples

In a *if-else* statement:

```
if($('Admin.Interview.exportLog.destination').empty().value()) {
  // true
} else {
  // false
}
```

With Geo value types, value() returns arrays of decimals:

```
// Point as a array of decimals: longitude and latitude
$('Point').value()
// Point longitude
$('Point').value()[0]
// Point latitude
$('Point').value()[1]

// LineString as a array of points (i.e. array of array of decimals)
$('LineString').value()
// Latitude of the first point in the line
$('LineString').value()[0][1]

// Polygon as a array of lines (i.e. array of array of array of decimals)
$('Polygon').value()
// Number of lines in the polygon
$('Polygon').value().length
// Number of points in the first line of the polygon
$('Polygon').value()[0].length
// First point of the first line (array of decimals: longitude and
latitude)
$('Polygon').value()[0][0]
```

See Also

isNull, type

value type method

value

Returns or changes the value's type. This conversion can be destructive (lose precision) or impossible (convert 'ABC' to an integer). When the conversion is impossible, the script's evaluation fails.

> Improvement to consider: when conversion fails return null instead of throwing an exception.

Method of: Value

Syntax

```
type([name])
```

Parameters

name (optional) is the type name into which the value shall be converted. Possible values are (case insensitive): `text`, `integer`, `decimal`, `boolean`, `locale`, `datetime`, `date`, `binary` (see Value Types description).

Examples

Get the value's type:

```
$('SOME_DECIMAL').type().eq('decimal')
```

Change a value's type:

```
$('SOME_DECIMAL').type('integer')
```

See Also

map, value

length method

length

Returns the length of the value. The length of a value has different meanings depending on the value type:

- if value type is *binary* the length is the number of bytes of the value,
- else the length is number of characters of the string representation of the value.

Method of: Value

Syntax

```
length()
```

Parameters

No parameters

Examples

Get the binary value's length in bytes:

```
$('BINARY').length()
```

See Also

map, value

Value Sequence Methods

append method

append

Appends (adds at the end) a value to a value to produce a value sequence. Also accepts a value sequence as input, in which case, both sequences are concatenated to produce a single one (it does not produce a sequence of sequence).

If the value being added is not of the same type as the sequence, it will be converted to the sequence's type. If the conversion fails, an exception is thrown.

If the sequence is `null` or empty, this method returns a new sequence containing the parameter(s) (this part is different from push). If the parameter is `null`, a `null` value is appended.

Method of: Value Sequence

Syntax

```
append(value [,value])
```

Parameters

value: the value to append to the sequence.

Examples

```
// Add a value to a sequence, then compute the average of the resulting sequence
$('BloodPressure:Measure.RES_PULSE').append($('StandingHeight:FIRST_RES_PULSE')).avg();

// Append several values to a value to produce the sequence: a, b, c, null
newValue('a').append('b', 'c', null);

// Append several values to a value sequence to produce the sequence: a, b, c, d, null
newSequence(['a', 'b']).append('c', 'd', null);
```

See Also

first, last, valueAt, size, sort, sum, avg, push, prepend

asSequence method

asSequence

Turns a value object into a value sequence object. A value sequence *is-a* value object but also *has-some* value objects (see Value Sequence Methods). If the value object on which it is applied is already a value sequence object, no operation is made.

Method of: Value Sequence

Syntax

```
asSequence()
```

Parameters

No parameters.

Examples

Create some value sequences:

```
// Creates a value sequence of type 'text' with one item
newValue('lorem ipsum').asSequence()

// Make sure that the value returned by $group is always a value sequence
$group('Var1','key','Var2').asSequence()
```

See Also

newValue, newSequence

avg method

avg

Returns the average of a value sequence.

If the sequence isn't a numerical sequence, magma exception is thrown.

If the sequence is `null` or empty, `null` is returned. Each `null` value of the sequence is turned into `0`. This method also accepts a non-sequence value, in which case, it is returned untouched.

Method of: Value Sequence

Syntax

```
avg()
```

Parameters

No parameters.

Examples

```
$('StandingHeight:Measure.RES_HEIGHT_MEASURE').avg();
```

See Also

first, last, valueAt, size, sort, sum, push

filter method

filter

Filter values of a sequence using a custom javascript predicating function.

Method of: Value Sequence

Syntax

```
filter(predicate)
```

Parameters

predicate predicating javascript function called for each value (with arguments: value and index of the value in the sequence) and that returns a boolean value (true if value is to be kept).

Examples

Filter sequence values being greater than a given value:

```
$('SequenceVar').filter(function(value) {
  return value.ge(100);
})
```

Filter sequence values based on value position in the sequence:

```
$('SequenceVar').filter(function(value,index) {
  return value.ge(100).value() && index<3;
})
```

Filter sequence based on another variable (in the same occurrence group) value at the same position:

```
var timepoint = $('TimePoint')
$('Measure').filter(function(value,index) {
  return timepoint.valueAt(index).isNull().not();
})
```

See Also

first, last, valueAt, size, trimmer, subset, map, reduce

first method

first

Returns the first value in a value sequence.

Method of: Value Sequence

Syntax

```
first()
```

Parameters

No parameters.

Examples

```
$('Admin.StageInstance.stage').first()
```

See Also

empty, last, valueAt, size

firstNotNull method

firstNotNull

Returns the first Value object of the sequence which value is not null. If the value sequence is empty or if there aren't not null values in the sequence a Value object with a null value is returned.

Method of: Value Sequence

Syntax

```
firstNotNull()
```

Parameters

No parameters.

Examples

```
$('Admin.StageInstance.stage').firstNotNull()
```

See Also

empty, first, last, valueAt, size

indexOf method

indexOf

Returns the first position of a value in a value sequence.

Method of: Value Sequence

Syntax

```
indexOf(value)
```

Parameters

value a primitive value or a value object to be looked for.

Examples

```
$('Admin.StageInstance.stage').indexOf('Spirometry')
```

See Also

empty, first, valueAt, size, lastIndexOf

insertAt method

insertAt

Insert at a given position a value to a value to produce a value sequence. Also accepts a value sequence as input, in which case, both sequences are concatenated to produce a single one (it does not produce a sequence of sequence).

If the value being added is not of the same type as the sequence, it will be converted to the sequence's type. If the conversion fails, an exception is thrown.

If the sequence is `null` or empty, this method returns a new sequence containing the parameter(s) (this part is different from push). If the parameter is `null`, a `null` value is appended.

Method of: Value Sequence

Syntax

```
insertAt(position, value [,value])
```

Parameters

position: the position (0-based) at which the value is to be inserted. If this position is greater than the original value sequence length, `null` values will be inserted as well until the requested position.

value: the value to append to the sequence.

Examples

```
// Add a value to a sequence, then compute the average of the resulting
sequence
$('BloodPressure:Measure.RES_PULSE').instertAt(0,
$('StandingHeight:FIRST_RES_PULSE')).avg();

// Insert several values to a value to produce the sequence: a, b, c, null
newValue('a').insertAt(1, 'b', 'c', null);

// Insert several values to a value to produce the sequence: a, null, b, c,
null
newValue('a').insertAt(2, 'b', 'c', null);

// Insert several values to a value sequence to produce the sequence: a, c,
d, null, b
newSequence(['a', 'b']).insertAt(1, 'c', 'd', null);
```

See Also

first, last, valueAt, size, sort, sum, avg, push, prepend, append

isSequence method

isSequence

Check whether a value is a value sequence object.

Method of: Value Sequence

Syntax

```
isSequence()
```

Parameters

No parameters.

Examples

Create some value sequences:

```
// Would be True if the VAR is a repeatable variable or if VAR is in
different tables referred by the view
$('VAR').isSequence()
```

See Also

newValue, newSequence

join method

join

Joins the text representation of the values in the sequence, using the provided delimiter, prefix and suffix. A `null` (resp. empty sequence) will return a `null` (resp. empty) text value.

Method of: Value Sequence

Syntax

```
join([delimiter[, prefix[, suffix]]])
```

Parameters

delimiter the delimiter text (string or value) to be used to join values of the sequence.

prefix the prefix text (string or value) to be added at the beginning of the resulting text (will not be applied if empty).

suffix the suffix text (string or value) to be added at the end of the resulting text (will not be applied if empty).

Examples

Join the value sequence: `[1,2,3]`

```
// returns "1, 2, 3"
$('SequenceVar').join(', ');
// returns "[1:2:3]"
$('SequenceVar').join(':', '[', ']');
// returns "123"
$('SequenceVar').join();
```

See Also

zip

last method

last

Returns the last value in a value sequence.

Method of: Value Sequence

Syntax

```
last()
```

Parameters

No parameters.

Examples

```
$('Admin.StageInstance.stage').last()
```

See Also

empty, first, valueAt, size

lastIndexOf method

lastIndexOf

Returns the last position of a value in a value sequence.

Method of: Value Sequence

Syntax

```
lastIndexOf(value)
```

Parameters

value a primitive value or a value object to be looked for.

Examples

```
$('Admin.StageInstance.stage').lastIndexOf('Spirometry')
```

See Also

empty, first, valueAt, size, indexOf

max method

avg

Returns the maximum value of a value sequence.

If the sequence isn't a numerical sequence, magma exception is thrown.

If the sequence is `null` or empty, `null` is returned. Each `null` value of the sequence is ignored. This method also accepts a non-sequence value, in which case, it is returned untouched.

Method of: Value Sequence

Syntax

```
max()
```

Parameters

No parameters.

Examples

```
$('StandingHeight:Measure.RES_HEIGHT_MEASURE').max();
```

See Also

first, last, valueAt, size, sort, sum, push

min method

avg

Returns the minimum value of a value sequence.

If the sequence isn't a numerical sequence, magma exception is thrown.

If the sequence is `null` or empty, `null` is returned. Each `null` value of the sequence is ignored. This method also accepts a non-sequence value, in which case, it is returned untouched.

Method of: Value Sequence

Syntax

```
min()
```

Parameters

No parameters.

Examples

```
$('StandingHeight:Measure.RES_HEIGHT_MEASURE').min();
```

See Also

first, last, valueAt, size, sort, sum, push

prepend method

prepend

Prepends (adds at the beginning) a value to a value to produce a value sequence. Also accepts a value sequence as input, in which case, both sequences are concatenated to produce a single one (it does not produce a sequence of sequence).

If the value being added is not of the same type as the sequence, it will be converted to the sequence's type. If the conversion fails, an exception is thrown.

If the sequence is `null` or empty, this method returns a new sequence containing the parameter(s). If the parameter is `null`, a `null` value is appended.

Method of: Value Sequence

Syntax

```
prepend(value [,value])
```

Parameters

value: the value to prepend to the sequence.

Examples

```
// Add a value to a sequence, then compute the average of the resulting
sequence
$('BloodPressure:Measure.RES_PULSE').prepend($('StandingHeight:FIRST_RES_P
ULSE')).avg();

// Prepend several values to a value to produce the sequence: b, c, null, a
newValue('a').prepend('b', 'c', null);

// Prepend several values to a value sequence to produce the sequence: c,
d, null, a, b
newSequence(['a', 'b']).prepend('c', 'd', null);
```

See Also

first, last, valueAt, size, sort, sum, avg, push, append

push method

push

Adds (at the end) a value to a value to produce a value sequence. Also accepts a value sequence as input, in which case, both sequences are concatenated to produce a single one (it does not produce a sequence of sequence).

If the value being added is not of the same type as the sequence, it will be converted to the sequence's type. If the conversion fails, an exception is thrown.

If the sequence is `null`, this method returns a null sequence (this part is different from append). If the sequence is empty, this method returns a new sequence containing the parameter(s). If the parameter is `null`, a `null` value is appended.

Method of: Value Sequence

Syntax

```
push(value [,value])
```

Parameters

value: the value to append to the sequence.

Examples

```
// Add a value to a sequence, then compute the average of the resulting
sequence
$('BloodPressure:Measure.RES_PULSE').push($('StandingHeight:FIRST_RES_PULS
E')).avg();
```

See Also

first, last, valueAt, size, sort, sum, avg, append

reduce method

reduce

Reduce values of a sequence to a single value using a custom javascript accumulating function. The reduction result is the last accumulated value.

Method of: Value Sequence

Syntax

```
reduce(accumulator[,initialValue])
```

Parameters

accumulator accumulating javascript function called for each value (with arguments: accumulated value, value and index of the value in the sequence) and that returns the new accumulated value. Note that if the accumulated value resulting of the function call is null, it is not applied to the final reduction result.

initialValue optional value to be used to initialize the accumulated value. If not provided, the accumulated value will be initialized with the first not null value of the sequence (thus in this case the accumulating function is not called during this initialization phase).

Examples

Reduce as sequence of measures to their sum:

```
$('Words').reduce(function(acc, value, index) {
  return acc.plus(value);
})
```

Reduce a sequence of measures to their average value initialized to 0:

```
$('SequenceVar').reduce(function(acc, value, index) {
  return acc.multiply(index).plus(value).div(index+1);
}, 0)
```

See Also

first, last, valueAt, size, trimmer, subset, map

size method

size

Returns the number of values within a sequence. Returns null value if operand is a null value.

Method of: Value Sequence

Syntax

```
size()
```

Parameters

No parameters.

Examples

```
$('Admin.StageInstance.stage').size()
```

See Also

empty, first, last, valueAt

sort method

sort

Sorts a sequence in natural order of its values or using a custom javascript comparing function.

Method of: Value Sequence

Syntax

```
sort([function])
```

Parameters

function optional javascript comparing function.

Examples

```
$('Admin.StageInstance.stage').sort()
```

```
$('Admin.StageInstance.stage').sort().first().any('MyStage')
```

Sorts a sequence of Datetime values according to their "dayOfYear" value:

```
$('Admin.Action.dateTime').sort(function(first, second) {
  // Custom sort method is expected to return a number
  return first.dayOfYear().value() - second.dayOfYear().value();
})
```

See Also

first, last, valueAt, size

stddev method

sum

Returns the standard deviation of a value sequence.

If the sequence isn't a numerical sequence, magma exception is thrown.

If the sequence is empty, `0` is returned. If the sequence is `null`, `null` is returned. Each `null` value of the sequence is turned into `0`. This method also accepts a non-sequence value, in which case, it is returned untouched.

Method of: Value Sequence

Syntax

```
stddev()
```

Parameters

No parameters.

Examples

```
$('StandingHeight:Measure.RES_HEIGHT_MEASURE').stddev();
```

See Also

first, last, valueAt, size, sort, avg, push

subset method

subset

Filter values of a sequence by subsetting the values according to their position.

Method of: Value Sequence

Syntax

```
subset(from[,to])
```

Parameters

from 0-based index (inclusive) from which value should be included.

to optional 0-based index (exclusive) to which value should be included

Examples

Subset sequence values from a position and its equivalent filter:

```
$('SequenceVar').subset(1)

// is equivalent to the filter:
$('SequenceVar').filter(function(value,index) {
  return index>=1;
})
```

Subset sequence values in a position range and its equivalent filter:

```
$('SequenceVar').subset(1,4)

// is equivalent to the filter:
$('SequenceVar').filter(function(value,index) {
  return index>=1 && index<4;
})
```

See Also

first, last, valueAt, size, filter

sum method

sum

Returns the sum of a value sequence. The returned value will have the operands value type (summing an `integer` sequence produces an `integer`).

If the sequence isn't a numerical sequence, magma exception is thrown.

If the sequence is empty, `0` is returned. If the sequence is `null`, `null` is returned. Each `null` value of the sequence is turned into `0`. This method also accepts a non-sequence value, in which case, it is returned untouched.

Method of: Value Sequence

Syntax

```
sum()
```

Parameters

No parameters.

Examples

```
$('StandingHeight:Measure.RES_HEIGHT_MEASURE').sum();
```

See Also

first, last, valueAt, size, sort, avg, push

trimmer method

trimmer

Filter values of a sequence by removing null values.

Method of: Value Sequence

Syntax

```
trimmer()
```

Parameters

No parameters.

Examples

Trim sequence values and its equivalent filter:

```
$('SequenceVar').trimmer()

// is equivalent to the filter:
$('SequenceVar').filter(function(value) {
  return value.isNull().not();
})
```

See Also

first, last, valueAt, size, filter

valueAt method

valueAt

Returns the a value at a specified index within the sequence (0-based).

Method of: Value Sequence

Syntax

```
valueAt(index)
```

Parameters

index is the 0-based index of the value to be retrieved from the sequence.

Examples

```
$('Admin.StageInstance.stage').valueAt(4)
```

When a view refers to several tables and a variable with same name exists in these tables, the resulting value is a sequence (see $ method). The order of the values in this sequence is the same as the order of the tables in the view. Then *valueAt* can be used to pick the value of a specitific table.

```
// variable VAR is defined in tables T1 and T2 referred by the view
// getting the value of table T2 by position
$('VAR').valueAt(1)
// is equivalent to fully qualifying the variable
$('project.T2.VAR')
```

See Also

first, last, size, sort

zip method

zip

Returns a sequence of values, where each value is the transformation of a tuple of values. The i-th tuple contains the i-th element from each of the argument sequences. The returned list length is the length of the longest argument sequence (shortest argument sequence values are null). Not sequential arguments have their value repeated in each tuple.

Method of: Value Sequence

Syntax

```
zip(value_1[, value_2[, ...]]], function)
```

Parameters

value\i the value(s) to be zipped to.

function the transformation function to be applied to each tuple of values.

Examples

Zip the value sequences:

- `[A,B,C]`
- `[1,2,3]`
- `[X,Y]`

And concatenate the string representation of the values for each tuples.

```
// returns the text values sequence ["A: 1X","B: 2Y","C: 3null"]
$('SequenceVarABC').zip($('SequenceVar123'),$('SequenceVarXY'),function(o1,
o2, o3) {
    return o1.concat(': ', o2, o3);
  });
```

Increment each values of a sequence of integers: `[1,2,3]`

```
// returns the integer values sequence [2,3,4]
$('SequenceVar123').zip(1,function(o1, o2) {
    return o1.plus(o2);
  });
```

See Also

join

Boolean Value Methods

- and method
- compare method
- eq method
- or method

and method

and

Applies the ternary and logic on values. If no arguments is provided, returns the value of the left operand.

Method of: Boolean Value

Syntax

```
and(param_1[, param_i[, ...]])
```

Parameters

param_i a boolean `Value` to be compared to.

Examples

```
$('BooleanVar').and($('OtherBooleanVar'))
```

```
$('BooleanVar').and($('OtherBooleanVar').not())
```

```
$('BooleanVar').and($('SomeBooleanVar'), $('OtherBooleanVar'))
```

See Also

not, or, isNull

compare method

compare

When comparing Boolean values: returns 0 if the value represents the same boolean value as the argument; a positive integer if the value represents `true` and the argument represents `false`; and a negative integer if this value represents `false` and the argument represents `true`.

When comparing Numeric values (i.e. `integer` and/or `decimal` types) or Text values: returns a negative integer, zero, or a positive integer as the value is less than, equal to, or greater than the value argument.

Method of: Boolean Value, Numeric Value, Text Value

Syntax

```
compare(value)
```

Parameters

value the `Value` to be compared to.

Examples

```
$('AVar').compare($('OtherVar'))
```

See Also

eq, compareNoCase

eq method

eq

Returns if left operand value is equal to right operand value. The operands must be either be both of:

- integer or decimal type.
- boolean type.
- text type.

If the left operand is a value sequence, the method will check equality for each of the values provided (sequences must be of same length and content must be equal). See also any to check if one of the value is in the value sequence.

Method of: Boolean Value, Numeric Value, Text Value

Syntax

```
eq(value [,value])
```

Parameters

value the value to be compared to, can be a primitive type or a `Value` object.

Examples

```
$('AVar').eq($('OtherVar'));

// Check sequence
$('VAR1').eq('a','b');
```

See Also

and, compare, ge, gt, le, lt, not, or, isNull

or method

or

Applies the ternary or logic on values. If no arguments is provided, returns the value of the left operand.

Method of: Boolean Value

Syntax

```
or(param_1[, param_i[, ...]])
```

Parameters

param_i a boolean `Value` to be compared to.

Examples

```
$('BooleanVar').or($('OtherBooleanVar'))
```

```
$('BooleanVar').or($('OtherBooleanVar').not())
```

```
$('BooleanVar').or($('SomeBooleanVar'), $('OtherBooleanVar'))
```

See Also

and, not, isNull

Numeric Value Methods

abs method

abs

Returns the absolute value of the current value.

Method of: Numeric Value

Syntax

```
abs()
```

Examples

```
$('AVar').abs()
```

See Also

ln

div method

div

Returns result of first operand value divided by second operand value. The operands must be either be of integer or decimal type. The result of the div operations is always of decimal type.

Method of: Numeric Value

Syntax

```
div(value)
```

Parameters

value the right operand `Value`.

Examples

```
$('AVar').div($('OtherVar'))
```

See Also

plus, minus, multiply

ge method

ge

Returns if left operand value is greater equal than right operand value. The operands must be either be of integer or decimal type.

Method of: Numeric Value

Syntax

```
ge(value)
```

Parameters

value the `Value` to be compared to.

Examples

```
$('AVar').ge($('OtherVar'))
```

See Also

eq, gt, le, lt

group method

group

Groups values from a continuous space into a discrete space given a list of adjacent range limits. Applies only to integer or decimal type values. The returned value is:

- the text representation of the range, for instance:
  - '-10' for range (-10),
  - '10-20' for range [10..20),
  - '20+' for range [20..+).
- the text representation of the value if the value is defined as an outlier.

Method of: Numeric Value

Syntax

```
group(array_of_bounds[, array_of_outliers])
```

Parameters

`array_of_bounds`: a list of values that will be the bounds of the ranges.

`array_of_outliers`: (optional) a list of outlier values that will be not be grouped and will be returned as is.

Examples

```
// usage example, possible returned values are: '-18', '18-35', '35-40',
..., '70+'
 $('CURRENT_AGE').group([18,35,40,45,50,55,60,65,70]);

// support of optional outliers
$('CURRENT_AGE').group([18,35,40,45,50,55,60,65,70],[888,999]);

// in combination with map
$('CURRENT_AGE').group([30,40,50,60],[888,999]).map({
   '-30' :  1,
   '30-40': 2,
   '40-50': 3,
   '50-60': 4,
   '60+':   5,
   '888':   88,
   '999':   99
 });
```

See Also

map

gt method

gt

Returns if left operand value is greater than right operand value. The operands must be either be of integer or decimal type.

Method of: Numeric Value

Syntax

```
gt(value)
```

Parameters

value the `Value` to be compared to.

Examples

```
$('AVar').gt($('OtherVar'))
```

See Also

eq, ge, le, lt

le method

le

Returns if left operand value is lower equal than right operand value. The operands must be either be of integer or decimal type.

Method of: Numeric Value

Syntax

```
le(value)
```

Parameters

value the `Value` to be compared to.

Examples

```
$('AVar').le($('OtherVar'))
```

See Also

eq, ge, gt, lt

ln method

ln

Returns the natural logarithm (base e) of the value. To obtain other logarithms, use the `log()` or `log(base)` methods (see below)

Method of: Numeric Value

Syntax

```
ln() // base e logarithm
log() // base 10 logarithm
log(base)  // arbitrary base logarithm
```

Parameters

base the base of the logarithm to evaluate

Examples

```
$('AVar').ln()
$('AVar').log(2) // base-2 logarithm
```

See Also

abs, root, pow

lt method

lt

Returns if left operand value is lower than right operand value. The operands must be either be of integer or decimal type.

Method of: Numeric Value

Syntax

```
lt(value)
```

Parameters

value the `Value` to be compared to.

Examples

```
$('AVar').lt($('OtherVar'))
```

See Also

eq, ge, gt, le

minus method

minus

Returns result of first operand value minus second operand value. The operands must be either of integer or decimal type.

Method of: Numeric Value

Syntax

```
minus(value)
```

Parameters

value the right operand `Value`.

Examples

```
$('AVar').minus($('OtherVar'))
```

See Also

plus, multiply, div

multiply method

multiply

Returns result of first operand value multiply second operand value. The operands must be either of integer or decimal type.

Method of: Numeric Value

Syntax

```
multiply(value)
```

Parameters

value the right operand `Value`.

Examples

```
$('AVar').multiply($('OtherVar'))
```

See Also

plus, minus, div

plus method

plus

Returns result of first operand value plus second operand value. The operands must be either of integer or decimal type.

Method of: Numeric Value

Syntax

```
plus(value)
```

Parameters

value the right operand `Value`.

Examples

```
$('AVar').plus($('OtherVar'))
```

See Also

minus, multiply, div

pow method

pow

Returns the value raised to the power of the argument.

Method of: Numeric Value

Syntax

```
pow(power)
```

Parameters

power the power to raise the value to.

Examples

```
$('AVar').pow(2) // AVar * AVar
$('AVar').pow(-1) // 1 / AVar
```

See Also

abs, ln, root

root method

sqroot, cbroot, root

Returns the (square, cubic or other) root of the value.

Method of: Numeric Value

Syntax

```
sqroot() // the square root
cbroot() // the cubic root
root(root) // arbitrary root
```

Parameters

root the root to evaluate.

Examples

```
$('AVar').sqroot()
$('AVar').root(2) // same as sqroot()
```

See Also

abs, ln, pow

round method

round

Returns the rounded value of the current value.

Method of: Numeric Value

Syntax

```
round()
round(scale)
```

Parameters

scale the number of decimals, default is 2.

Examples

```
// round to 2 decimals (default)
$('AVar').round()
// round to specified number of decimals
$('AVar').round(4)
```

See Also

abs

Measurement Unit Methods

- toUnit method
- unit method

toUnit method

toUnit

Converts the current value with a measurment unit to another measurement unit. For example, this method can convert the value 1kg to 2.2lb or 1000g.

Method of: Measurement Unit

Syntax

```
toUnit(newUnit) // converts the current value to a new measurment unit
```

Parameters

newUnit the measurement unit to convert to (must be a string).

Examples

```
$('HEIGHT').unit('m').toUnit('cm') // converts the value from meters to
centimeters
$('HEIGHT').toUnit('cm') // converts the value from its current unit to
centimeters (the current unit is take from the HEIGHT variable).
```

See Also

unit

unit method

unit

Sets or gets the measurement unit. When called without any arguments, this method returns the current measurement unit of the value. When called with a string argument, this method sets the measurement unit of the current value, regardless of the previous measurement unit (if any).

Method of: Measurement Unit

Syntax

```
unit() // returns the current measurement unit (text)
unit(value) // sets the current measurement unit to 'value' (must be a string)
```

Parameters

value the new measurement unit (must be a string).

Examples

```
$('HEIGHT').unit() // returns 'cm'
$('HEIGHT').unit('cm') // returns the same value, but with a measurement unit of 'cm'
```

See Also

toUnit

Text Value Methods

- capitalize method
- compareNoCase method
- concat method
- date method
- datetime method
- lowerCase method
- matches method
- replace method
- trim method
- upperCase method

capitalize method

capitalize

Returns a copy of the string, with first characters of each word capitalized.

Method of: Text Value

Syntax

```
capitalize([delimiters])
```

Parameters

delimiters (optional) the delimiting characters for identifying words. Default is ' '.

Examples

```
$('AVar').capitalize()
$('AVar').capitalize(':;,( .["')
```

See Also

concat, lowerCase, replace, trim, upperCase

compareNoCase method

compareNoCase

Returns a negative integer, zero, or a positive integer as the text value is less than, equal to, or greater than the text value argument ignoring case.

Method of: Text Value

Syntax

```
compareNoCase(value)
```

Parameters

value the text `Value` to be compared to ignoring case.

Examples

```
$('AVar').compareNoCase($('OtherVar'))
```

See Also

eq, compare

concat method

concat

Returns the text type result of first operand concat second operand. The operands must be either values or text type.

Method of: Text Value

Syntax

```
concat(value_1[, value_i[, ...]])
```

Parameters

value_i the value which string representation is to be concatenated to the text value.

Examples

```
$('AVar').concat($('OtherVar'))
```

See Also

capitalize, lowerCase, replace, trim, upperCase

date method

date

Makes a value of date type by parsing the text value given a date format pattern.

The pattern should be defined from the Java Date Format Specifications:

Letter	Date or time component	Example
G	Era designator	AD
y	Year	1996; 96
Y	Week year	2009; 09
M	Month in year	July; Jul; 07
w	Week in year	27
W	Week in month	2
D	Day in year	189
d	Day in month	10
F	Day of week in month	2
E	Day name in week	Tuesday; Tue
u	Day number of week (1 = Monday, ..., 7 = Sunday)	1
a	Am/pm marker	PM
H	Hour in day (0-23)	0
k	Hour in day (1-24)	24
K	Hour in am/pm (0-11)	0
h	Hour in am/pm (1-12)	12
m	Minute in hour	30
s	Second in minute	55
S	Millisecond	978
z	Time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	-0800
X	Time zone	-08; -0800; -08:00

Method of: Text Value

Syntax

```
date(format)
```

Parameters

format the date format pattern to be used.

Examples

```
// example: 08/23/73
$('AVar').date('MM/dd/yy')

// example: Dec 31, 2009
$('AVar').date('MMM dd, yyyy')

// example: Wed, Jul 4, '01
$('AVar').date("EEE, MMM d, ''yy")
```

See Also

datetime

datetime method

datetime

Makes a value of datetime type by parsing the text value given a date format pattern.

The pattern should be defined from the Java Date Format Specifications:

Letter	Date or time component	Example
G	Era designator	AD
y	Year	1996; 96
Y	Week year	2009; 09
M	Month in year	July; Jul; 07
w	Week in year	27
W	Week in month	2
D	Day in year	189
d	Day in month	10
F	Day of week in month	2
E	Day name in week	Tuesday; Tue
u	Day number of week (1 = Monday, ..., 7 = Sunday)	1
a	Am/pm marker	PM
H	Hour in day (0-23)	0
k	Hour in day (1-24)	24
K	Hour in am/pm (0-11)	0
h	Hour in am/pm (1-12)	12

m	Minute in hour	30
s	Second in minute	55
S	Millisecond	978
z	Time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	-0800
X	Time zone	-08; -0800; -08:00

Method of: Text Value

Syntax

```
datetime(format)
```

Parameters

format the date format pattern to be used.

Examples

```
// example: 08/23/73
$('AVar').datetime('MM/dd/yy')

// example: 10/23/12 10:59 PM
$('AVar').date('MM/dd/yy h:mm a')

// example: Wed, 4 Jul 2001 12:08:56 -0700
$('AVar').date('EEE, d MMM yyyy HH:mm:ss Z')

// example: 2008-03-01T13:00:00+01:00
$('AVar').date("yyyy-MM-dd'T'HH:mm:ssZ")
```

See Also

date

lowerCase method

lowerCase

Returns a copy of the string, lower cased.

Method of: Text Value

Syntax

```
lowerCase([locale])
```

Parameters

locale (optional) the locale to be used, as a string with syntax: `language[_country[_variant]]`.

Examples

```
$('AVar').lowerCase()
$('AVar').lowerCase('fr_CA')
```

See Also

capitalize, concat, replace, trim, upperCase

matches method

matches

Returns a Boolean value after match of a regular expression against a string. See Regular Expressions in JavaScript Guide for more details about how to write a regular expression pattern.

Method of: Text Value

Syntax

```
matches(regex)
```

Parameters

regex the regular expression to be searched for.

Examples

```
$('VarName').matches(/yes/)
```

See Also

eq, compare, compareNoCase

replace method

replace

Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.

See javascript replace.

Method of: Text Value

Syntax

```
replace(regex, text)
```

Parameters

regex the regular expression to be searched for.

text the string that replaces the matched substring.

Examples

```
$('VarName').replace('regex', '$1')
```

See Also

capitalize, concat, lowerCase, trim, upperCase

trim method

trim

Returns a copy of the string, with leading and trailing whitespace omitted.

Method of: Text Value

Syntax

```
trim()
```

Parameters

No parameters.

Examples

```
$('AVar').trim()
```

See Also

capitalize, concat, lowerCase, replace, upperCase

upperCase method

upperCase

Returns a copy of the string, upper cased.

Method of: Text Value

Syntax

```
upperCase([locale])
```

Parameters

locale (optional) the locale to be used, as a string with syntax: `language[_country[_variant]]`.

Examples

```
$('AVar').upperCase()
$('AVar').upperCase('fr_CA')
```

See Also

capitalize, concat, lowerCase, replace, trim

Date and Date Time Value Methods

- add method
- after method
- before method
- dayOfMonth method
- dayOfWeek method
- dayOfYear method
- format method
- hour method
- hourOfDay method
- millisecond method
- minute method
- month method
- quarter method
- second method
- semester method
- time method
- weekday method
- weekend method
- weekOfMonth method
- weekOfYear method
- year method

add method

add

Adds days to a value of date or date time type.

Method of: Date and Date Time Value

Syntax

```
add(days)
```

Parameters

days the number of days to be added.

Examples

Adds 2 days:

```
$('Date').add(2)
```

Subtracts 4 days:

```
$('Date').add(-4)
```

after method

after

Returns true if the date value is after the specified date value(s).

Method of: Date and Date Time Value

Syntax

```
after(date_1[, date_2[, ...]])
```

Parameters

date_i the dates to be compared to.

Examples

After one date:

```
$('Date').after($('OtherDate'))
// string representation of dates are supported as well
$('Date').after('2017-01-15')
```

After several dates:

```
$('Date').after($('OtherDate'), $('SomeOtherDate'))
```

before method

after

Returns true if the date value is before the specified date value(s).

Method of: Date and Date Time Value

Syntax

```
before(date_1[, date_2[, ...]])
```

Parameters

date_i the dates to be compared to.

Examples

Before one date:

```
$('Date').before($('OtherDate'))
// string representation of dates are supported as well
$('Date').before('2017-01-15')
```

Before several dates:

```
$('Date').before($('OtherDate'), $('SomeOtherDate'))
```

dayOfMonth method

dayOfMonth

Returns the day of month from a date as an integer starting from 1.

Method of: Date and Date Time Value

Syntax

```
dayOfMonth()
```

Parameters

No parameters.

Examples

```
$('Date').dayOfMonth()
```

See Also

dayOfWeek, dayOfYear

dayOfWeek method

dayOfWeek

Returns the day of week from a date as an integer starting from 1 (Sunday).

Method of: Date and Date Time Value

Syntax

```
dayOfWeek()
```

Parameters

No parameters.

Examples

```
$('Date').dayOfWeek()
```

See Also

dayOfMonth, dayOfYear

dayOfYear method

dayOfYear

Returns the day of year from a date as an integer starting from 1.

Method of: Date and Date Time Value

Syntax

```
dayOfYear()
```

Parameters

No parameters.

Examples

```
$('Date').dayOfYear()
```

See Also

dayOfMonth, dayOfWeek

format method

format

Returns the text representation of the date formatted by the provided pattern.

Date and time formats are specified by date and time pattern strings. The following pattern letters are defined:

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10

F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800

Pattern letters are usually repeated, as their number determines the exact presentation:

- Text: if the number of pattern letters is 4 or more, the full form is used; otherwise a short or abbreviated form is used if available.
- Number: the number of pattern letters is the minimum number of digits, and shorter numbers are zero-padded to this amount.
- Year: if the number of pattern letters is 2, the year is truncated to 2 digits; otherwise it is interpreted as a number.
- Month: if the number of pattern letters is 3 or more, the month is interpreted as text; otherwise, it is interpreted as a number.

The following examples show how date and time patterns are interpreted in the U.S. locale. The given date and time are 2001-07-04 12:08:56 local time in the U.S. Pacific Time time zone.

Date and Time Pattern	Result
'yyyy.MM.dd G, HH:mm:ss z'	2001.07.04 AD, 12:08:56 PDT
'EEE, MMM d, yy'	Wed, Jul 4, 01
'h:mm a'	12:08 PM
'K:mm a, z'	0:08 PM, PDT
'yyyyy.MMMMM.dd GGG hh:mm aaa'	02001.July.04 AD 12:08 PM
'EEE, d MMM yyyy HH:mm:ss Z'	Wed, 4 Jul 2001 12:08:56 -0700
'yyMMddHHmmssZ'	010704120856-0700
'yyyy-MM-dd'T'HH:mm:ss.SSSZ'	2001-07-04T12:08:56.235-0700

Method of: Date and Date Time Value

Syntax

```
format(pattern)
```

Parameters

pattern the pattern describing the date and time format.

Examples

String pattern:

```
$('Date').format('dd/MM/yyyy')
```

Pattern extracted from the string representation of a variable value:

```
$('Date').format($('Pattern'))
```

hour method

hour

Returns the hour of the day for the 12-hour clock (0 - 11). Noon and midnight are represented by 0, not by 12. For example, at 10:04:15.250 PM the HOUR is 10.

Method of: Date and Date Time Value

Syntax

```
hour()
```

Parameters

No parameters.

Examples

```
$('Date').hour()
```

See Also

year, month, hourOfDay, minute, second, millisecond

hourOfDay method

hourOfDay

Returns the hour of the day for the 24-hour clock. For example, at 10:04:15.250 PM the hour of the day is 22.

Method of: Date and Date Time Value

Syntax

```
hourOfDay()
```

Parameters

No parameters.

Examples

```
$('Date').hourOfDay()
```

See Also

year, month, hour, minute, second, millisecond

millisecond method

millisecond

Returns the millisecond within the second.

Method of: Date and Date Time Value

Syntax

```
millisecond()
```

Parameters

No parameters.

Examples

```
$('Date').millisecond()
```

See Also

year, month, hour, hourOfDay, minute, second

minute method

minute

Returns the minute within the hour.

Method of: Date and Date Time Value

Syntax

```
minute()
```

Parameters

No parameters.

Examples

```
$('Date').minute()
```

See Also

year, month, hour, hourOfDay, second, millisecond

month method

month

Returns the month of a Date as an integer starting from 0 (January).

Method of: Date and Date Time Value

Syntax

```
month()
```

Parameters

No parameters.

Examples

```
$('Date').month()
```

See Also

year, hour, hourOfDay, minute, second, millisecond

quarter method

quarter

Returns the quarter of a Date as an integer starting from 0 (Q1).

Method of: Date and Date Time Value

Syntax

```
quarter()
```

Parameters

No parameters.

Examples

```
$('Date').quarter()
```

See Also

year, semester, month, hour, hourOfDay, minute, second, millisecond

second method

second

Returns the second within the minute.

Method of: Date and Date Time Value

Syntax

```
second()
```

Parameters

No parameters.

Examples

```
$('Date').second()
```

See Also

year, month, hour, hourOfDay, minute, millisecond

semester method

semester

Returns the semester of a Date as an integer starting from 0 (Q1).

Method of: Date and Date Time Value

Syntax

```
semester()
```

Parameters

No parameters.

Examples

```
$('Date').semester()
```

See Also

year, quarter, month, hour, hourOfDay, minute, second, millisecond

time method

time

Returns the number of milliseconds since January 1, 1970, 00:00:00 GMT (epoch time).

Method of: Date and Date Time Value

Syntax

```
time()
```

Parameters

No parameters.

Examples

```
$('Date').time()
```

See Also

year, month, hour, hourOfDay, millisecond, minute, second

weekday method

weekday

Returns a boolean value indicating whether the date denotes a weekday (between Monday and Friday inclusively).

Method of: Date and Date Time Value

Syntax

```
weekday()
```

Parameters

No parameters.

Examples

```
$('Date').weekday()
```

See Also

weekend

weekend method

weekend

Returns a boolean value indicating whether the date denotes a weekend (either Sunday or Saturday).

Method of: Date and Date Time Value

Syntax

```
weekend()
```

Parameters

No parameters.

Examples

```
$('Date').weekend()
```

See Also

weekday

weekOfMonth method

weekOfMonth

Returns the week of month from a date as an integer starting from 1.

Method of: Date and Date Time Value

Syntax

```
weekOfMonth()
```

Parameters

No parameters.

Examples

```
$('Date').weekOfMonth()
```

See Also

weekOfYear

weekOfYear method

weekOfYear

Returns the week of year from a date as an integer starting from 1.

Method of: Date and Date Time Value

Syntax

```
weekOfYear()
```

Parameters

No parameters.

Examples

```
$('Date').weekOfYear()
```

See Also

weekOfMonth

year method

year

Returns the year value.

Method of: Date and Date Time Value

Syntax

```
year()
```

Parameters

No parameters.

Examples

```
$('Date').year()
```

See Also

month, hour, hourOfDay, minute, second, millisecond

Geo Value Methods

- latitude method
- longitude method

latitude method

latitude

Returns the latitude of a *point* value as a *decimal* value.

Method of: Geo Value

Syntax

```
latitude()
```

Parameters

No parameters.

Examples

```
$('COORDINATE').latitude()
```

See Also

longitude

longitude method

longitude

Returns the longitude of a *point* value as a *decimal* value.

Method of: Geo Value

Syntax

```
longitude()
```

Parameters

No parameters.

Examples

```
$('COORDINATE').longitude()
```

See Also

latitude