Opal Documentation

OBiBa

Jul 02, 2019

Contents

Targeted at individual studies and study consortia, OBiBa software stack (Opal, Mica etc.) provides a software solution for epidemiological data management, analysis and publication. Opal is the core data warehouse application that provides all the necessary tools to import, transform and describe data. Opal can be used with Agate, the OBiBa's central authentication server.

Warning: Opal documentation is in the process of being rewritten. See also the `Opal Documentation Archive`

CHAPTER 1

Introduction

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.

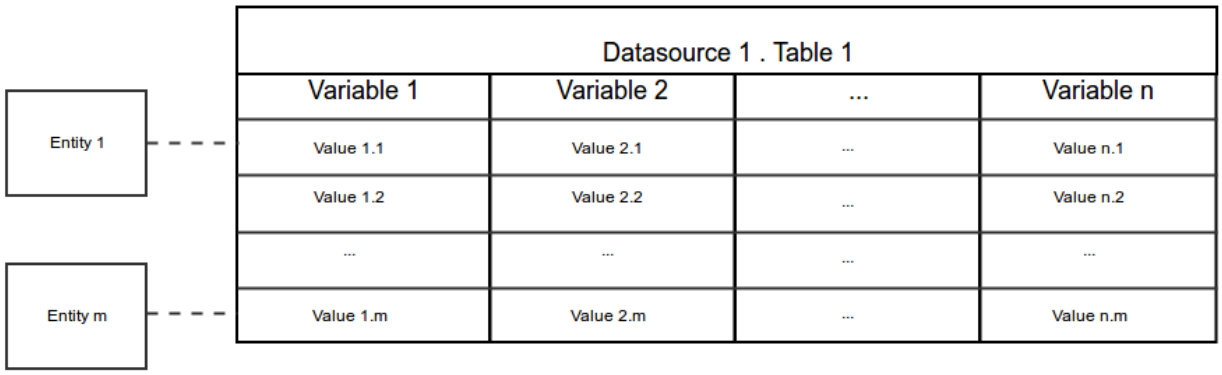
See detailed concepts and tutorials:

1.1 Variables and Data

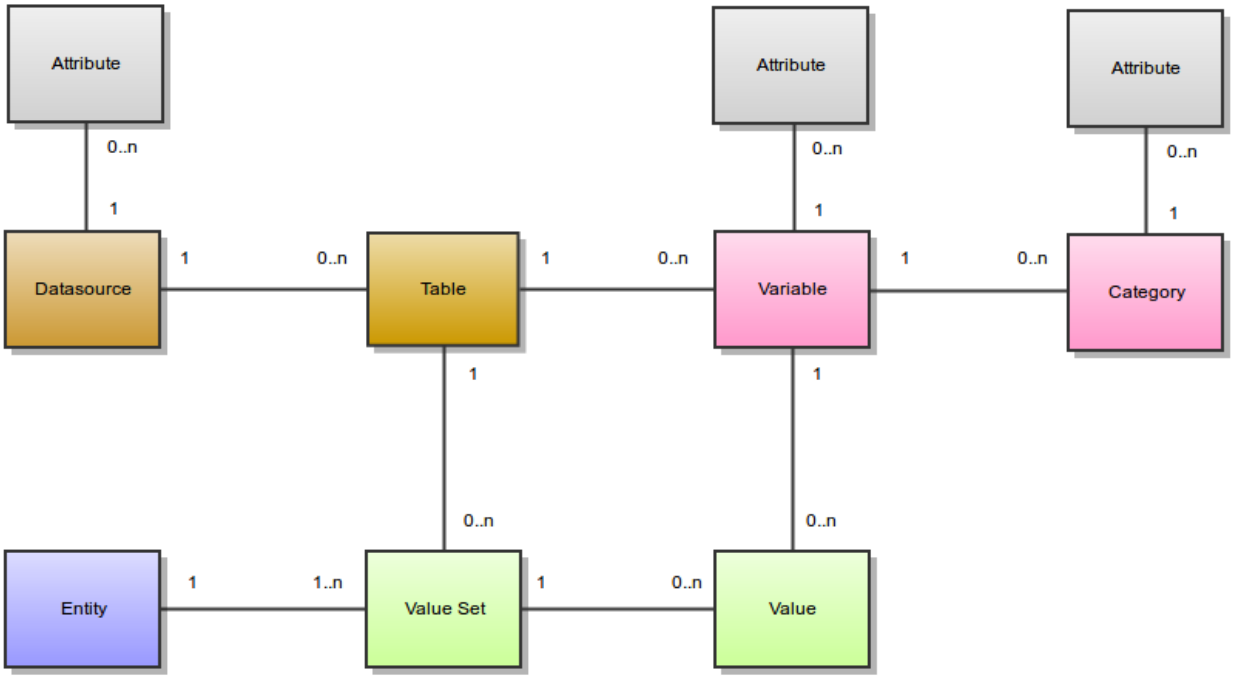
The variables are organized in an abstract way, independently of the way they are persisted.

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.



The following diagram shows the relationships between the different concepts:



1.1.1 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
- point
- linestring
- polygon

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 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 is useful for disambiguation.

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 table **Table1**:

ID	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**:

ID	Var1	Var3
1	Value 1.1	Value 3.1
3	Value 1.3	Value 3.3

or **View2**:

ID	DerivedVar = function(Var1, Var2)
1	function(Value1.1, Value2.1)
3	function(Value1.3, Value2.3)

Given Table1 above and the following table **Table2**:

ID	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, as in **View3**:

ID	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

1.1.2 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

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.

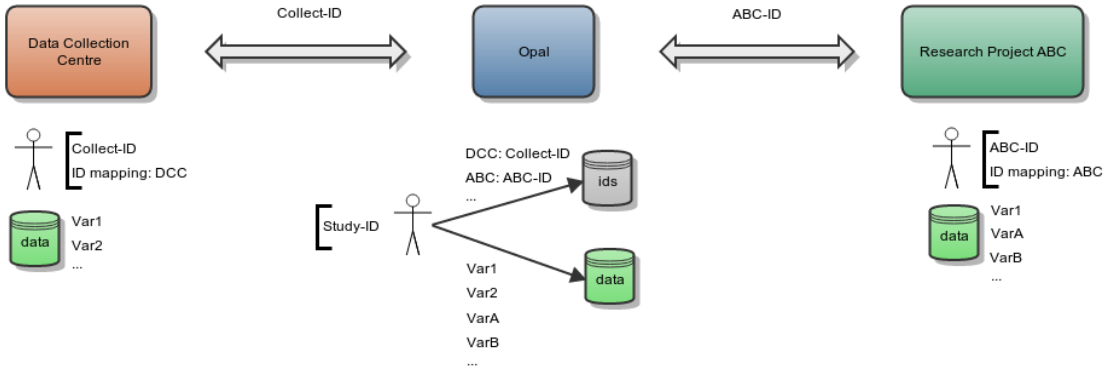
A value is always associated with a type and a data (or a sequence of data if the variable is `repeatable`).

1.2 Identifiers Mappings

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).

1.2.1 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.

1.2.2 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.

1.3 Data Harmonization

1.3.1 Opal Application

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

Data Harmonization with Opal

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

- Set up Opal servers for each Biobanks and import relevant data sets,
- Configure a harmonized description of data in each Opal server,
- Set up a Mica server that is able to authenticate itself against each Opal server,
- 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.

1.3.2 Data Harmonization Infrastructure

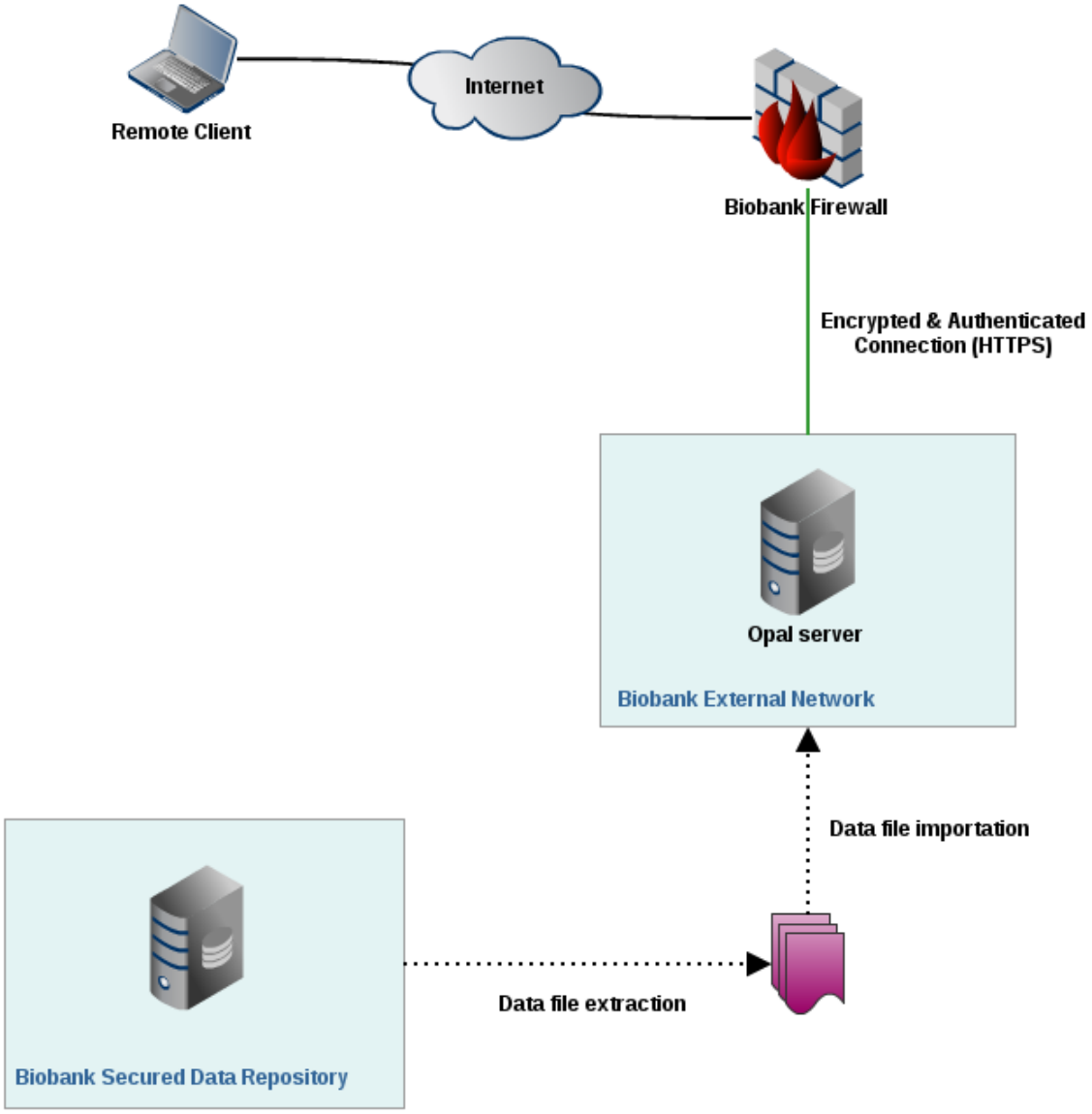
Opal in Study Networks

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:

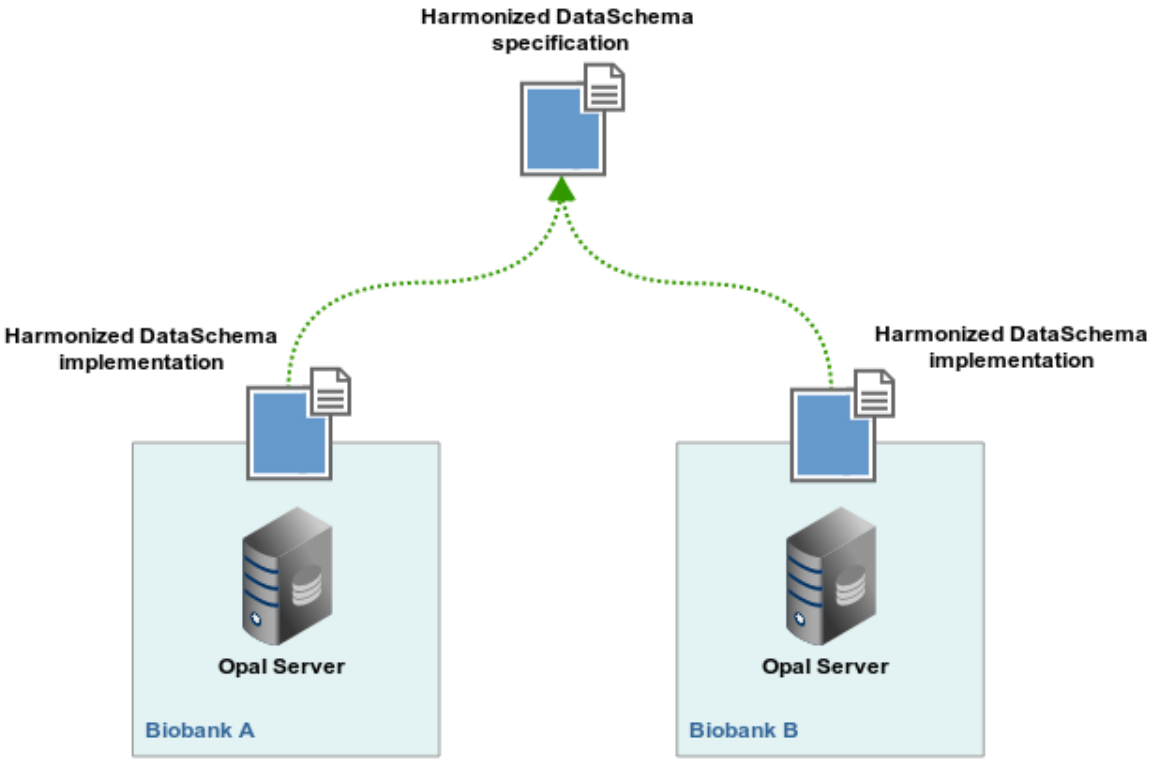
- The Opal server is hosted in a network that is protected by a firewall which only allows connections using encrypted (HTTPS) protocol through a specific port. Connections can also be restricted to specific remote clients.
- Opal application itself requires user authentication. Data hosted by Opal are 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 Studies

The aim of the data harmonization process is to transform study-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 study-specific variables into DataSchema format variables.

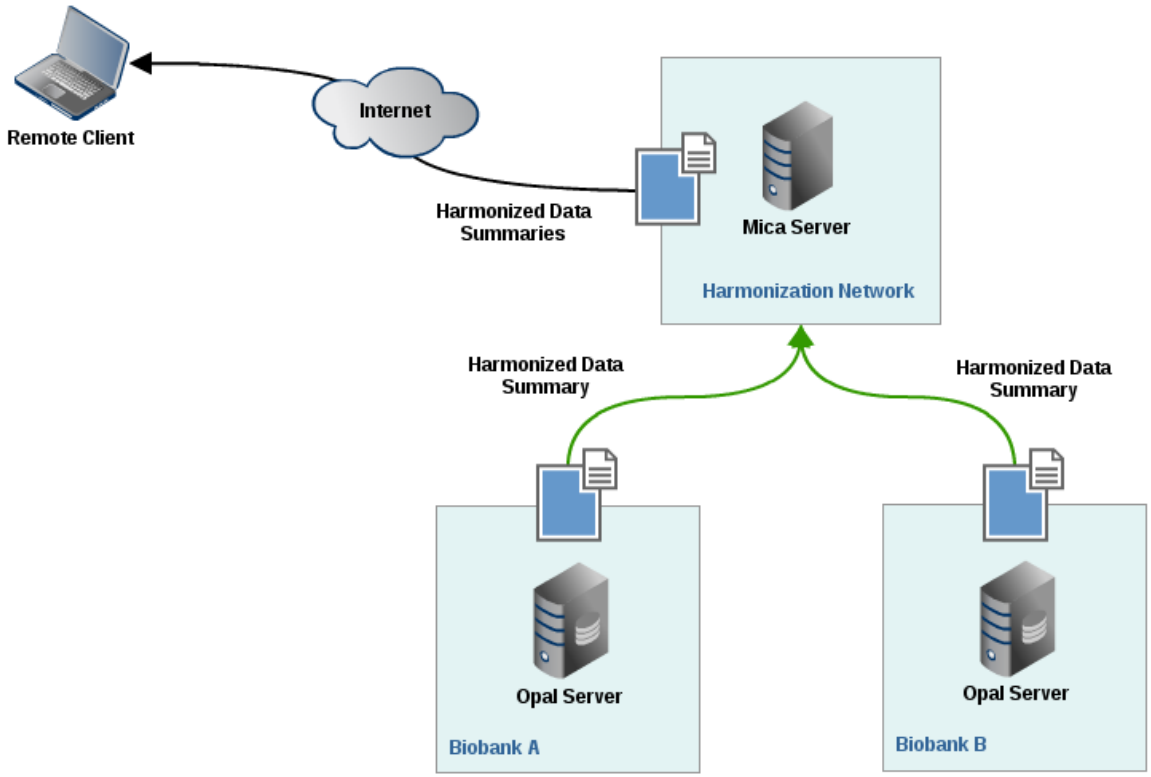


Study Consortium Web Portal

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.

CHAPTER 2

Installation

Opal is a stand-alone Java server application that does not require a database engine at installation time. Connection to one or more databases is part of the post-install configuration.

2.1 Requirements

2.1.1 Server Hardware Requirements

Component	Requirement
CPU	Recent server-grade or high-end consumer-grade processor
Disk space	8GB or more (data are stored within the database, not in Opal server space).
Memory (RAM)	Minimum: 4GB, Recommended: >8GB

2.1.2 Server Software Requirements

Java is the minimum software requirement, other software are for a fully functional system. While Java is required by Opal server application, MongoDB, MySQL, R can be installed on another server. See also *R Server*.

Software	Suggested version	Download link	Usage
Java	= 1.8.x	Java Oracle downloads	Java runtime environment
MySQL	>= 5.5.x	MySQL downloads	Database engine
MongoDB	>= 2.4.x	MongoDB downloads	Database engine
R	>= 3.0.x	R downloads	Statistical analysis engine

2.2 Install

Opal is distributed as a Debian/RPM package and as a zip file. The resulting installation has default configuration that makes Opal ready to be used. Once installation is done, see *Configuration* instructions.

2.2.1 Debian Package Installation

Opal is available as a Debian package from OBiBa Debian repository. To proceed installation, do as follows:

- Install Debian package. Follow the instructions in the repository main page for installing Opal.
- Manage Opal Service: after package installation, Opal server is running: see how to manage the Service.

2.2.2 RPM Package Installation

Opal is available as a RPM package from OBiBa RPM repository. To proceed installation, do as follows:

- Install RPM package. Follow the instructions in the RPM repository main page for installing Opal.
- Manage Opal Service: after package installation, Opal is running: see how to manage the Service.

2.2.3 Zip Distribution Installation

Opal is also available as a Zip file. To install Opal zip distribution, proceed as follows:

- Download Opal distribution
- Unzip the Opal distribution. Note that the zip file contains a root directory named **opal-x.y.z-dist** (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.
- Create an `OPAL_HOME` environment variable
- Separate Opal home from Opal distribution directories (recommended). This will facilitate subsequent upgrades.

Set-up example for Linux:

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

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

For the administrator accounts, the credentials are “administrator” as username and “password” as password. See User Directories Configuration to change it.

2.3 Upgrade

The upgrade procedures are handled by the application itself.

2.3.1 Debian Package Upgrade

If you installed Opal via the Debian package, you may update it using the command:

```
apt-get install opal
```

2.3.2 RPM Package Upgrade

If you installed Opal via the RPM package, you may update it using the command:

```
yum install opal-server
```

2.3.3 Zip Distribution Upgrade

Follow the Installation of Opal Zip distribution above but make sure you don't overwrite your opal-home directory.

2.4 Execution

2.4.1 Server launch

Service

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

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**.

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

```
service opal help
```

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.

Environment variable	Required	Description
`OPAL_HOME`	yes	Path to the Opal "home" 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`

Execute the command line (bin directory is in your execution PATH)):

```
opal
```

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.

2.4.2 Usage

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

- http://localhost:8080 will provide a connection without encryption,
- https://localhost:8443 will provide a connection secured with ssl.

2.4.3 Troubleshooting

If you encounter an issue during the installation and you can't resolve it, please report it in our Opal Issue Tracker.

Opal logs can be found in **/var/log/opal**. If the installation fails, always refer to this log when reporting an error.

CHAPTER 3

Configuration

3.1 Main Configuration File

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

3.1.1 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.

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.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.

3.1.2 SSH Server Configuration

Opal is accessible using SSH clients: SFTP is available through SSH connections. The SSH connection port can be configured.

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

3.1.3 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.

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).

3.1.4 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:

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).
`org.obiba.opal.r.repos`	The list of CRAN repositories from which R packages can be downloaded, comma separated. Default value is `https://cran.rstudio.com,https://cran.obiba.org`.

3.1.5 Login Policy Configuration

To prevent brute force password guessing, a user can be temporarily banned after too many login failures.

Property	Description
`org.obiba.opal.security.login.maxRetry`	Number of failed login attempts before being banned (default is `3`).
`org.obiba.opal.security.login.retryTime`	Time span in which the maximum of retry count should happen before starting a ban period, in seconds (default is `300`). No time limit if not positive.
`org.obiba.opal.security.login.banTime`	Ban time after max retry, within the retry time span, was reached, in seconds (default is `300`). No ban if not positive.

3.1.6 Agate Server Configuration

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

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`.

3.1.7 System Identifiers Generation Configuration

When importing data and selecting a identifiers mapping, if an imported identifier does not exist for the selected mapping and the strategy that was chosen is to generate a system identifier, then the following default settings apply for system identifiers generation:

Property	Description
`org.obiba.opal.identifiers.length`	Length of the numerical part of the identifier (i.e. not including the prefix length). Default is `10`.
`org.obiba.opal.identifiers.zeros`	Allow leading zeros in the numerical part of the identifiers. Default is `false`.
`org.obiba.opal.identifiers.prefix`	Character prefix to be applied. Default is none.
`org.obiba.opal.identifiers.checksum`	Add a checksum digit so that the generated identifier can be validated regarding the Luhn algorithm. Default is `false`.

3.1.8 Miscelaneous Configuration

Advanced settings.

Property	Description
`org.obiba.opal.keys.entityType`	Type of entities to store in the identifiers table.
`org.obiba.opal.keys.tableReference`	Fully-qualified name of the identifiers table
`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).
`org.obiba.opal.plugins.site`	The URL to the plugins repository (default is https://plugins.obiba.org). 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.
`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 data to a file (CSV, SAS, SPSS, Stata). If empty for the considered entity type, the default column name will apply. The format to be used is a comma-separated key-value list, for instance: `org.obiba.magma.entityIdNames=Participant=Idepic,Biomarker=Biom_Id`
`org.obiba.magma.entityIdName`	Specify the default column name to be used for the entity identifier when exporting data to a file (CSV, SAS, SPSS, Stata). If empty, this name depends on the file format.
`org.obiba.magma.readDataPointsCount`	Maximum number of data points (number of rows per number of variables) when batches of values are read from a table. Default value is `100000`.
`org.obiba.opal.security.multiProfile`	Allow user to login from different realms with the same username. Note that the user is always logged in one realm at a time (no addition of the privileges). Default value is `true`.
`org.obiba.opal.security.ssl.allowInvalidCertificates`	When connecting to MongoDB using SSL and when remote certificate is self-signed, the certificate check can be deactivated (not recommended, default is `false`).
`org.obiba.opal.jdbc.maxPoolSize`	Maximum size of the pool of JDBC connections, for each SQL database. Default value is `100`.

3.2 Advanced Configuration File

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

3.2.1 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:

```
<!-- 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:dirsomedir. 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:dirsomedir c:/dir/somedir

3.3 User Directories

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**. See also Shiro ini file documentation.

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

By default Opal server has several built-in user directories (in the world of Shiro, a user directory is called a realm):

- a file-based user directory (**shiro.ini** file),
- the internal Opal user directory,
- the user directory provided by Agate.

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**.

Note: It is not recommended to use this file-based user directory. It is mainly dedicated to define a default system super-user.

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.

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

(continues on next page)

(continued from previous page)

```
[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:
#   DEBIAN:
#     cd /usr/share/opal/tools && ./shiro-hasher -p
#   UNIX:
#     cd <OPAL_DIST_HOME>/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/
↪OEk0jsZbYXjiGhR7/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]*
opal-administrator = *
```

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

```
cd /usr/share/opal/tools
./shiro-hasher -p
```

3.3.1 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:

```
[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:

```
[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 =
```

3.3.2 Atlassian Crowd User Directory

Atlassian Crowd is not supported any more because the connector was based on libraries with security issues. OpenID Connect is to be preferred for authentication delegation. For more information see section *Identity Providers*.

3.3.3 Other Settings

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:

```
# =======================
# 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.

3.4 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,
- specifies acceptable protocols and cipher suites,
- refines 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>
```

For performance, you can also activate Apache's compression module (mod_deflate) with the following settings (note the json content type setting) in file */etc/apache2/mods-available/deflate.conf*:

```
<IfModule mod_deflate.c>
  <IfModule mod_filter.c>
      # these are known to be safe with MSIE 6
      AddOutputFilterByType DEFLATE text/html text/plain text/xml
      # everything else may cause problems with MSIE 6
      AddOutputFilterByType DEFLATE text/css
      AddOutputFilterByType DEFLATE application/x-javascript application/javascript␣
↪application/ecmascript
      AddOutputFilterByType DEFLATE application/rss+xml
      AddOutputFilterByType DEFLATE application/xml
      AddOutputFilterByType DEFLATE application/json
  </IfModule>
</IfModule>
```

CHAPTER 4

Plugins

4.1 Repository

Opal plugins available are:

Name	Type	Description	Depends	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
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.	htslib executables (bcftools, tabix, bgzip)	VCF Store Plugin API
opal-datasource-limesurvey	opal-datasource	Connects to a Limesurvey database to extract variables and data.	No dependencies	Datasource Plugin API
opal-datasource-redcap	opal-datasource	Connects to a REDCap server to extract variables and data.	No dependencies	Datasource Plugin API
opal-datasource-spss	opal-datasource	Java implementation of a SPSS file reader.	No dependencies	Datasource Plugin API
opal-datasource-readr	opal-datasource	R implementation of a rectangular format file reader, based on readr.	R server	Datasource Plugin API
opal-datasource-readxl	opal-datasource	R implementation of a Excel file reader/writer, based on readxl/writexl.	R server	Datasource Plugin API
opal-datasource-googlesheets4	opal-datasource	R implementation of a Google Sheets reader, based on googlesheets4.	R server	Datasource Plugin API
opal-datasource-validate	opal-analysis	R implementation of a data and metedata validator, based on validate.	R server	Analysis Plugin API

4.2 Installation

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

4.2.1 Automatic Installation

Because having a search engine is an absolute requirement, Opal server will check at startup that there is a plugin of type `opal-search` and if it's not the case, the latest version of the opal-search-es plugin (that applies to the current Opal server version) will be automatically downloaded and installed without needing a server restart. If for any reason this plugin cannot be automatically downloaded (network issue), the Opal start-up will fail and you will need to install the plugin manually.

4.2.2 Manual Installation

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

- Download the plugin of interest (zip file) from OBiBa Plugins Repository,
- 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.
- Read the installation instructions (if any) of the plugin to identify the system dependencies or any other information,
- Restart Opal.

4.3 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.

4.4 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.

CHAPTER 5

R Server

5.1 Installation

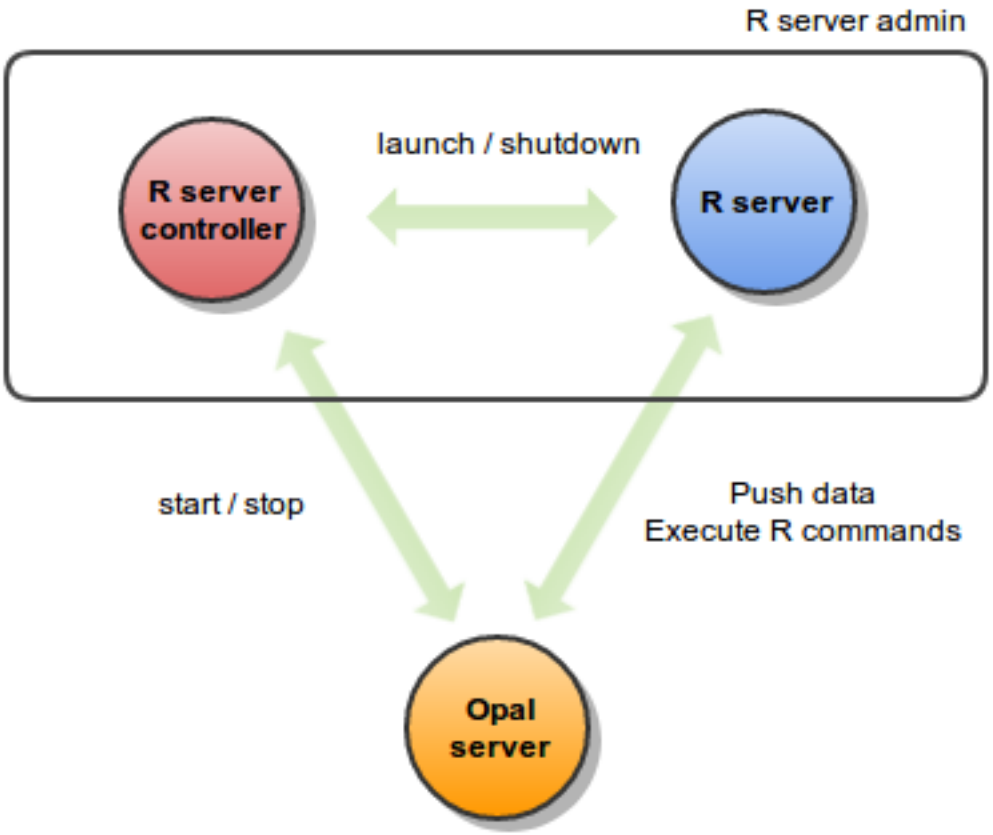
Opal is able to interact with a R server for running statistics analysis and reports.

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).



5.1.1 Requirements

Server Hardware Requirements

Component	Requirement
CPU	Recent server-grade or high-end consumer-grade processor
Disk space	8GB or more (data are stored within the database, not in Opal server space).
Memory (RAM)	Minimum: 4GB, Recommended: >8GB

Server Software Requirements

Java is the minimum software requirement, other software are for a fully functional system.

Software	Suggested version	Download link	Usage
Java	= 1.8.x	Java Oracle downloads	Java runtime environment
R	>= 3.0.x	R downloads	Statistical analysis engine

While Java is required by Opal server application, MongoDB/MySQL/R can be installed on another server.

5.1.2 Install

Opal is distributed as a Debian/RPM package and as a zip file. The resulting installation has default configuration that makes Opal ready to be used. Once installation is done, see *Configuration* instructions.

Debian Package Installation

Opal R Server is available as a Debian package from OBiBa Debian repository. To proceed installation, do as follows:

- Install Debian package. Follow the instructions in the repository main page for installing Opal.
- Manage Opal Service: after package installation, Opal server is running: see how to manage the Service.

RPM Package Installation

Opal R Server is available as a RPM package from OBiBa RPM repository. To proceed installation, do as follows:

- Install RPM package. Follow the instructions in the RPM repository main page for installing Opal.
- Manage Opal Service: after package installation, Opal is running: see how to manage the Service.

Zip Distribution Installation

Opal R Server is also available as a Zip file. To install Opal zip distribution, proceed as follows:

- Download R Server Admin distribution
- Unzip the R Server Admin distribution. Note that the zip file contains a root directory named **rserver-admin-x.y.z-dist** (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.
- Create a `RSERVER_HOME` environment variable
- Install Rserve, a R package that is library that enables the connection with R. This can be done within R by using the CRAN install command from the R console:

```
install.packages(c('rserve', 'opal', 'tidyverse', 'knitr', 'rmarkdown'), repos=c(
↪'http://cran.rstudio.com', 'http://cran.obiba.org'), dependencies=TRUE, lib='/usr/
↪local/lib/R/site-library')
```

5.1.3 Upgrade

The upgrade procedures are handled by the application itself.

Debian Package Upgrade

If you installed Opal via the Debian package, you may update it using the command:

```
apt-get install opal-rserver
```

RPM Package Upgrade

If you installed Opal via the RPM package, you may update it using the command:

```
yum install opal-rserver
```

Zip Distribution Upgrade

Follow the Installation of Opal Zip distribution above but make sure you don't overwrite your opal-home directory.

5.1.4 Execution

Server launch

Service

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

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**.

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

```
service rserver help
```

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

Manually

The R Server Admin application can be launched from the command line.

Environment variable	Required	Description
`RSERVER_HOME`	yes	Path to the R Server "home" directory.
`JAVA_OPTS`	no	Options for the Java Virtual Machine. For example: *-Xmx4096m -XX:MaxPermSize=256m*

To change the defaults update: `bin/rserver` or `bin/rserver.bat`

Execute the command line (bin directory is in your execution PATH)):

```
rserver
```

The R Server Admin server log files are located in **RSERVER_HOME/logs** directory. If the logs directory does not exist, it will be created by R Server.

Usage

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
```

Troubleshooting

If you encounter an issue during the installation and you can't resolve it, please report it in our Opal Issue Tracker.

Opal logs can be found in **/var/log/opal**. If the installation fails, always refer to this log when reporting an error.

Rserve logs can be found in **/var/lib/rserver/logs/Rserve.log** and might indicate R errors.

In case the **Rserve** R binary package does not match the R version, it is possible to update it from the R console (started as root), with the following command:

```
# install regular package
install.packages('Rserve', repos='http://cran.rstudio.com', lib='/usr/local/lib/R/
↪site-library')

# OR install package from source
install.packages('Rserve', 'http://www.rforge.net/', type='source', lib='/usr/local/
↪lib/R/site-library')
```

5.2 Configuration

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

5.2.1 Controller Configuration

The file **RSERVER_HOME/conf/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.

5.2.2 Rserve Configuration

The file **RSERVER_HOME/conf/Rserv.conf** allows the configuration of the core R server. See also the full documentation of the Rserv.conf 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.

5.2.3 R Session Configuration

When a new R session is started on server side the starting state of this session can be configured using the **RSERVER_HOME/conf/Rprofile.R**. Any R command (to be executed by the `rserver` user) can be put in this file.

CHAPTER 6

Web Introduction

The Opal Web Application is the web interface of the Opal server.

6.1 Requirements

This web interface is a javascript application requiring a modern web browser. There is no requirement regarding the operating system.

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

CHAPTER 7

Projects

7.1 Datasource Types

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		x	x	
Opal Archive Datasource		x	x	
SPSS Datasource	x	x		
SPSS R Datasource		x	x	
SAS R Datasource		x	x	
Stata R Datasource		x	x	
Excel Datasource	x			
Opal SQL Datasource				x
Tabular SQL Datasource		x	x	x
Limesurvey Datasource		x		
Opal Datasource		x		
MongoDB Datasource				x

7.1.1 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.

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: "

```
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.

7.1.2 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.

7.1.3 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.

7.1.4 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.

7.2 Project Tables

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'sdatabase) or a logical table (also called view) which is a set of derived variables (data are computed on-demand)

7.2.1 Operations

Download Dictionary

The whole project data dictionary can be download as an Excel file. This file is compatible with the operations of and Add/Update TablesAdd. When no tables are selected, the downloaded dictionary contains the definition of all tables. When some tables are selected, the dictionaryViewcontains 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, Oapl Archive), server-based (SQL, Limesurvey, Opal)
- Data format specific options
- Incremental options
- Identifiers mapping options
- Data dictionary update review and table to to import selection
- Data to import review
- Archiving options when dealing with a file-based datasource

Some import options can be describved 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 option, this allow to import small chunks of dataIncrementalat a time.
Identifier 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 acorresponding system identifier for the selected mapping • Ignore unknown identifiers at import: only data with identifier having a corresponding system identifier in the selected mappingwill be imported • Generate a system identifier for each unknown imported identifiers: a system identifier will be generated for each unknownimported identifier If no identifiers mapping is selected, the participant identifiers are imported as-is. Unknown participant identifiers will beautomatically added in Opal.

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), 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
- Identifier mapping options:
  - if an identifiers mapping is selected each entity to be exported nust have a mapped identifier. Otherwise the export will fail
  - if no identifiers is selected the data are exported with system identifiers

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 onlywhen one table is selected for copy).

Backup Views

Create an archived backup of views selection (or all views).

Add Table

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

Add/Update table from Dictionary

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 anexisting view by selecting "Download View XML") or an Excel file (see). If a view with same name already exists,Excel file templateconfirmation 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 variable algorithms are expressed using Magama Javascript API.

Add View

Restore Views

Restore backed up views. Restored views of the same name as those of existing views will be skipped unless the override options is checked.

Remove

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

7.3 Table Details

A table can be a raw table or a view

7.3.1 Dictionary

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

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

7.3.2 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)

7.3.3 Values

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

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

7.3.4 Permisions

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 does not grant access to individual-level data.

Administrate Table Permission

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

7.3.5 Operations

Add Variables to View

This operation consist of making a derived variable for each of the selected variables (see) and adding them to a view (either#Variable Selectionan 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 and Add View.

Export Data

Start Export Data procedure with the table preselected.

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

7.3.6 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*. An *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 an *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 criteria (for instance, all women older than 50 years). This scrip tmust return a logical value: *true*, the entity is kept, *false* (or *null*), it is excluded.

Variable Search

Variables can be searched. Selecting the suggested name goes to the corresponding variable details.

Variable List Filtering

The list of the variables can be filtered the same way the variables can be searched. On *ENTER* key pressed, the list is refred with all variables matching the criteria.

7.4 Variable Details

A variable describes the data.

7.4.1 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.

7.4.2 Properties

This section displays the proerties of the variable:

- Name
- Entity Type
- Value Type
- Repeatable
- Unit
- Referenced Entity Type
- Mime Type
- Occurence Group
- Index

7.4.3 Categories

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

- label (mapped *label* category attribute if this one is defined)
- missing (if the category indeicates a missing answer)

Edit Categories

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

7.4.4 Attributes

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

- namespace
- language
- value

Add Attributes

Thois operation addds a new attribute. The combination of namespace and name must be unique.

Edit

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

Remove

Remove the attributes.

7.4.5 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 missiong and non-missing values

7.4.6 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).

7.4.7 Values

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

7.4.8 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.

7.5 Querying for Variables

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

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

7.5.1 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.

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 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 values.

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 atttributes follow the same pattern, prefixed by `categorycategory-<namespace>-<name>-<locale>`.

The nature of the variables 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
```

CHAPTER 8

Search

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

8.1 Search Variables

Search for variables using facets and full-text query.

8.1.1 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	The variable field must be in at least one of the selected predefined values
Like	The variable field must match a query string (with wildcard support)
Not in	The negation of In.
Not like	The negation of 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.

8.1.2 Full-text Search

The full-text search applies to:

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

Wildcard can be used.

8.1.3 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.

8.1.4 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.

8.2 Search Entity

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

8.2.1 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.

8.3 Search Entities

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.

8.3.1 Prerequisite

Only tables which values have been indexed can be searched.

8.3.2 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:

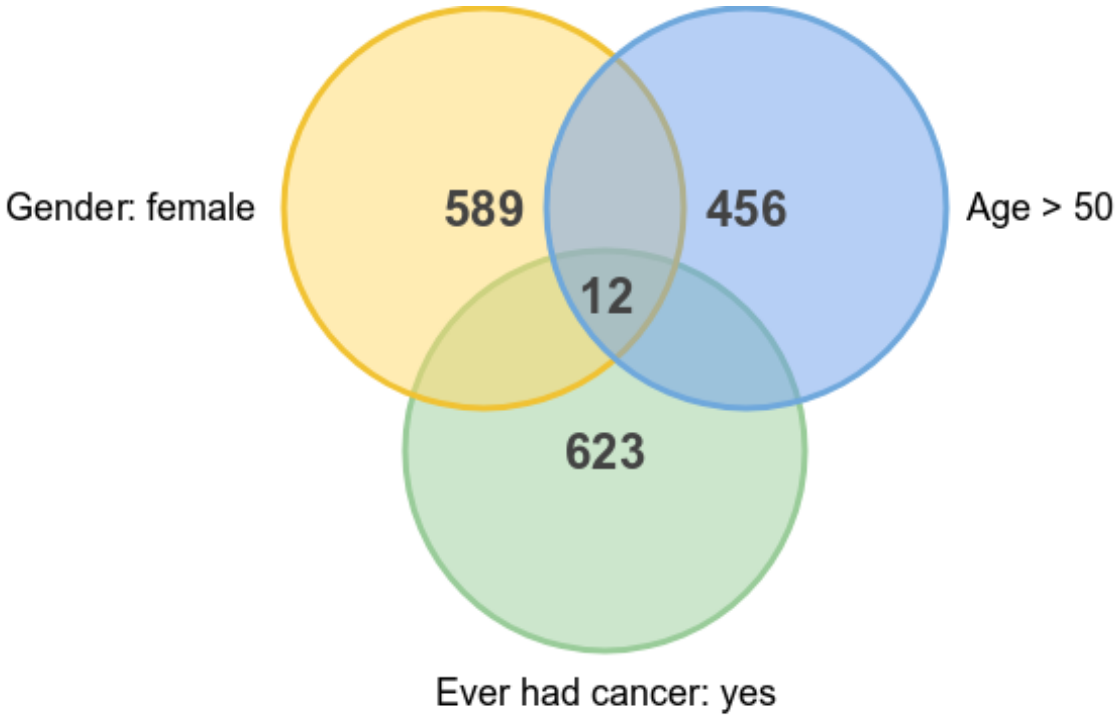
Operation	Description
All	The values of the variable can be any value (empty or not empty).
Empty	The values of the variable are empty (null value or value sequence with all null values).
Not empty	The values of the variable are not empty (not null value or value sequence with at least one not null value).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

8.3.3 Identifier Criterion

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

8.3.4 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.

CHAPTER 9

Administration

Administrate the Opal system services and settings.

9.1 General Settings

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

9.1.1 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.

9.1.2 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.

9.2 Identity Providers

9.2.1 Definition

Identity Providers are user registries that can be used to signin users into Opal. The OpenID Connect protocol is used to perform the authentication delegation: Opal has no access to user credentials, only basic user information is retrieved. An example of open source identity provider is Keycloak.

The only requirement is that the Identity Provider server exposes an OpenID Connect configuration discovery entry point. Usually it takes the form of: *https://auth.example.org/.well-known/openid-configuration* (see Google Accounts or ORCID examples).

9.2.2 Operations

Add ID Provider

Open a form dialog to specify the connection details to the ID provider.

There are some required fields:

- An ID provider must be identified by a *Name*,
- The Opal application has been registered in the this provider: these are the *Client ID* and *Client Secret* fields.
- The *Discovery URI* must follow the OpenID Connect configuration discovery specifications.

The optional fields are:

- The *Label* is a human-readable name that will be displayed in the provider's signin button in the login page. If missing, the name of the ID provider will be used.
- The *Account Login* address allows the user to go to it's personal profile page in the ID provider interface (to chenge its password for instance) from the Opal login page.
- The *Groups* are the group that are to be automatically applied to any users signing in through this ID provider.
- The *Scope* is the scope value(s) to be sent to the ID provider to initiate the OpenID Connect dialog. This is provider dependent but usually `openid` is enough.

Remove

Remove the ID provider. This does not affect the user profiles that may have been created through this provider and this does not remove the permissions specific to this provider that may have been applied.

Edit

Open the ID provider form dialog. Name cannot be changed (see *Duplicate* operation instead).

Enable

On creation an ID provider is disabled, which means that the corresponding Signin button is not shown in the login page.

Duplicate

Open an ID provider form dialog prefilled with the original provider's values (except for the name).

9.3 Taxonomies

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

9.3.1 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.

9.3.2 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 700K 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.

9.4 Databases

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, MariaDB, 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, MariaDB	Opal SQL	x		
MySQL, MariaDB, PostgreSQL	Tabular SQL	x	x	x
MySQL, MariaDB, PostgreSQL	Limesurvey		x	
MongoDB	Opal Documents	x		

9.4.1 Database Engines

SQL Databases

Currently the supported SQL database engines are: MySQL, MariaDB 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 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. Example of connection URLs: `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:

```
use admin
db.createUser(
  {
    user: "opaladmin",
    pwd: "opaladmin",
    roles: [
```

(continues on next page)

(continued from previous page)

```
      {
        "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.

Opal supports connection to MongoDB using SSL: add the `ssl=true` (and any other relevant parameters) to the MongoDB connection string. The system key-pair (see *Encryption Keys*) will be used for connecting to the database. If the MongoDB server certificate is self-signed, its certificate can be added to the Opal trusted certificates store by creating a Opal user authenticated by this certificate. See also usage of property `org.obiba.opal.security.ssl.allowInvalidCertificates` in *Miscelaneous Configuration*.

9.4.2 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.

9.4.3 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.

9.5 Plugins

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

9.5.1 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.

9.5.2 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).

9.5.3 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).

9.5.4 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 `org.obiba.opal.plugins.site` property in *Miscelaneous Configuration* instructions.

9.6 Java Virtual Machine

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 services.

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

CHAPTER 10

R DataSHIELD Introduction

10.1 What is R?

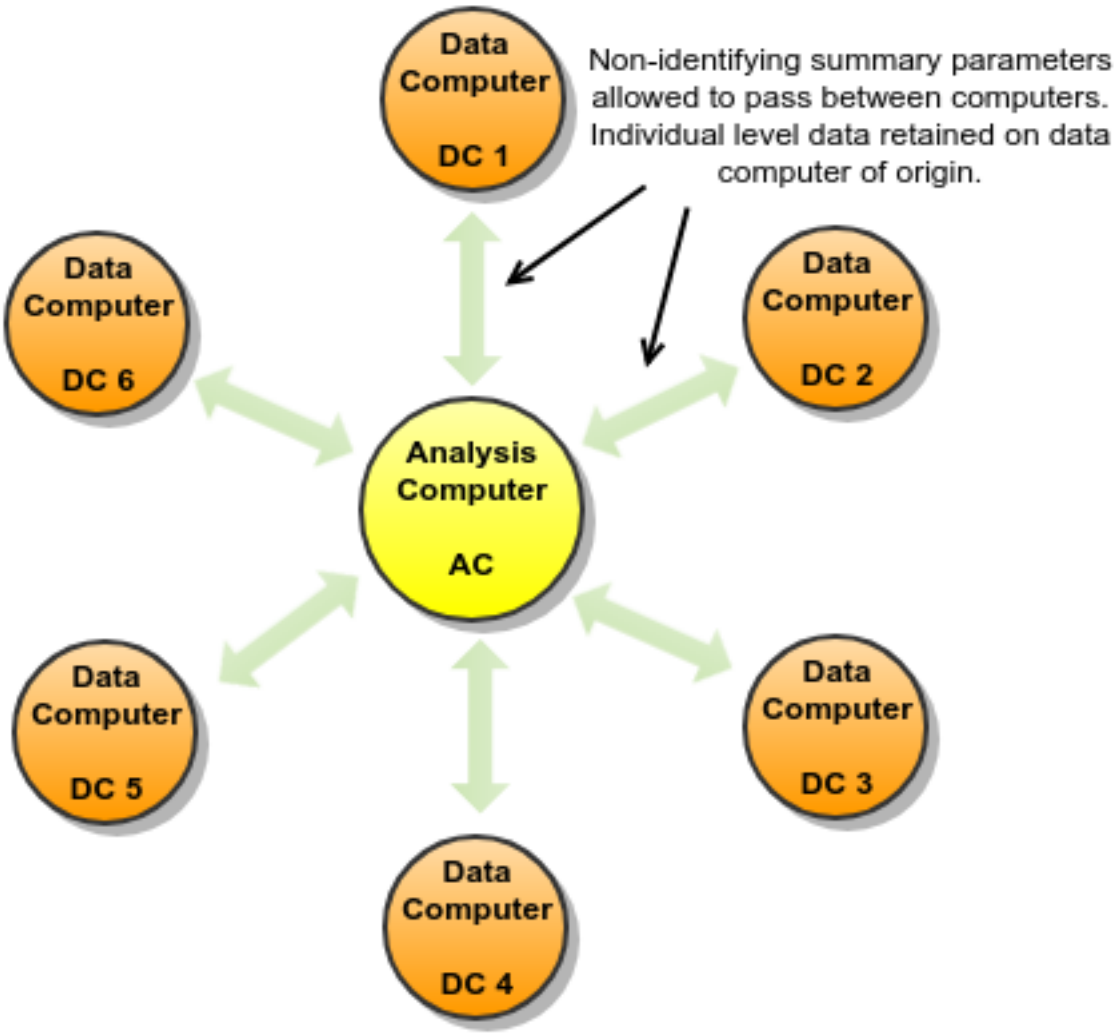
R is a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc.

Please consult the R project for further information.

10.2 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 following figure 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.

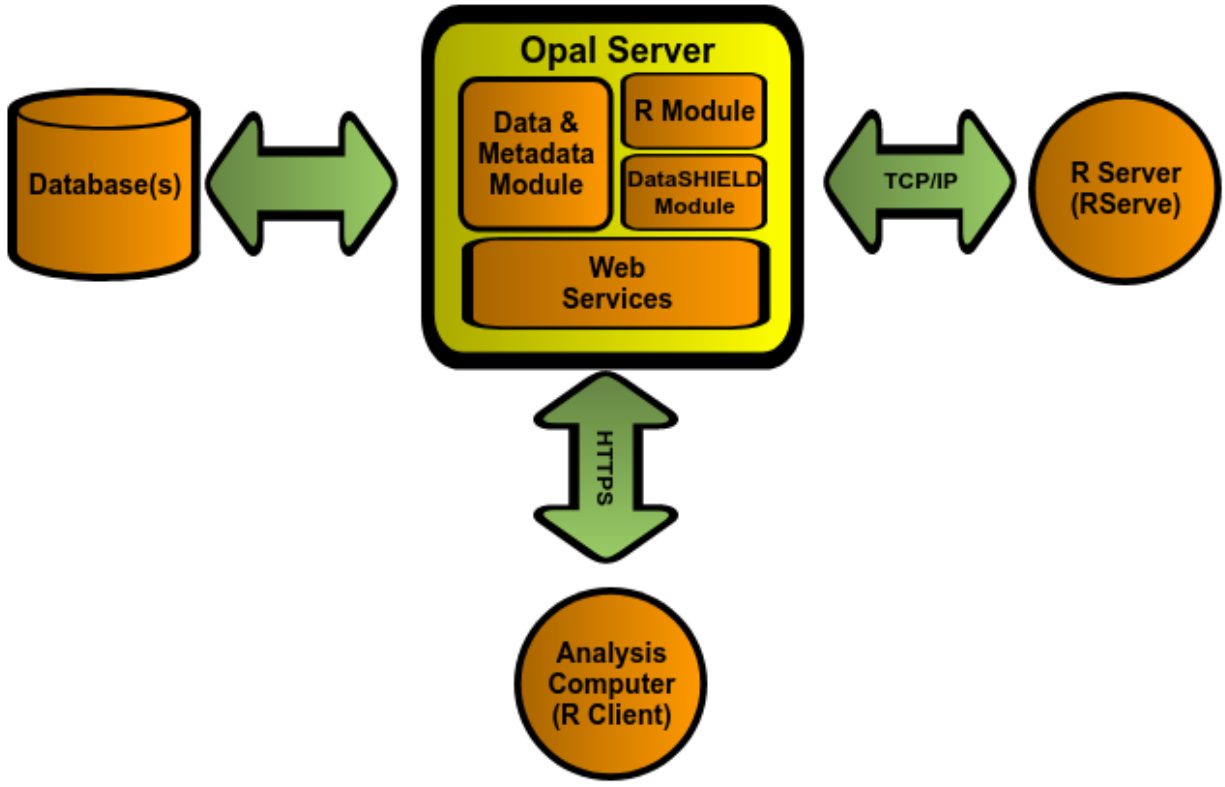


Please consult the DataSHIELD project for further information.

10.3 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)



10.3.1 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.

10.3.2 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.

10.3.3 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.

10.3.4 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.

10.3.5 Web Services

Interaction between these modules and their clients is done through Web Services.

10.3.6 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.

10.3.7 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.

CHAPTER 11

Using R

11.1 Prerequisites

On client side, R is to be available. See the R installation documentation that matches your system.

11.2 Installation

11.2.1 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('https://cran.rstudio.com/', 'https://cran.obiba.org
↪'), dependencies=TRUE)
```

11.2.2 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('https://cran.rstudio.com/', 'https://
↪www.stats.ox.ac.uk/pub/RWin/'))
```

Once these are installed, the opal package can be installed like so:

```
install.packages('opal', repos='https://cran.obiba.org', type='source')
```

11.3 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://opal-demo.obiba.org')

# assign some data to a data.frame
opal.assign(o,'CNSIM1','datashield.CNSIM1',variables=list('GENDER','PM_BMI_CONTINUOUS
↪'))

# do some analysis on the remote R session
opal.execute(o,'summary(CNSIM1)')

# get the remote data.frame on the client
D <- opal.execute(o,'CNSIM1')
head(D)

# clean remote R session
opal.logout(o)
```

See also Opal R example scripts.

11.4 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://opal-demo.obiba.org', opts=credentials)
```

CHAPTER 12

Using DataSHIELD

12.1 Prerequisites

On client side, R is to be available. See the R installation documentation that matches your system.

12.2 Installation

12.2.1 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('https://cran.rstudio.com/', 'https://cran.obiba.org
↪'), dependencies=TRUE)
```

12.2.2 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('https://cran.rstudio.com/', 'https://
↪www.stats.ox.ac.uk/pub/RWin/'))
```

Once these are installed, the opal package can be installed like so:

```
install.packages('opal', repos='https://cran.obiba.org', type='source')
```

12.3 Usage

12.3.1 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.

12.3.2 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 *datashieldclient* package is to be installed.

12.3.3 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'), 'https://cran.obiba.
↪org'), dependencies=TRUE)

# Load datashieldclient library
library(datashieldclient)
```

12.3.4 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.

12.3.5 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', 'https://opal.studya.org')
studyB <- opal.login('username', 'password', 'https://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', 'https://opal.studya.org')
studyB <- opal.login('username', 'password', 'https://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.

12.3.6 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:

- clients manipulate the server-side R environment (assign data, transform data, etc.)
- clients request an aggregate of some value in the R environment
- server extracts the requested value from the R environment
- server executes the aggregation method on the requested the data in a freshly created environment
- 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')
```

(continues on next page)

(continued from previous page)

```
# 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='https://opal.obiba.org',
                                                     S3='https://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))
```

12.4 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.

12.4.1 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.

12.4.2 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.

12.4.3 Contributing to DataSHIELD Packages

DataSHIELD packages sources are hosted on GitHub.

Some DataSHIELD developers documentation is also available.

CHAPTER 13

Reporting with R

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, free of charge (Open Source Edition) and is available as a Desktop or a Server application. The Server flavor is more suitable for teams (shared development environment) and when restrictive security constraints apply (IP white-listing).

13.1 Prerequisites

In order to be able to interact with a Opal server, the prerequisites are the following:

- having R installed both on client and on server sides
- having R package *opal* installed
- having access to a Opal server

13.2 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).

13.2.1 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
- analyze, 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.

13.2.2 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 the Opal Reporting with R output.

13.2.3 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.

13.2.4 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.

13.3 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.

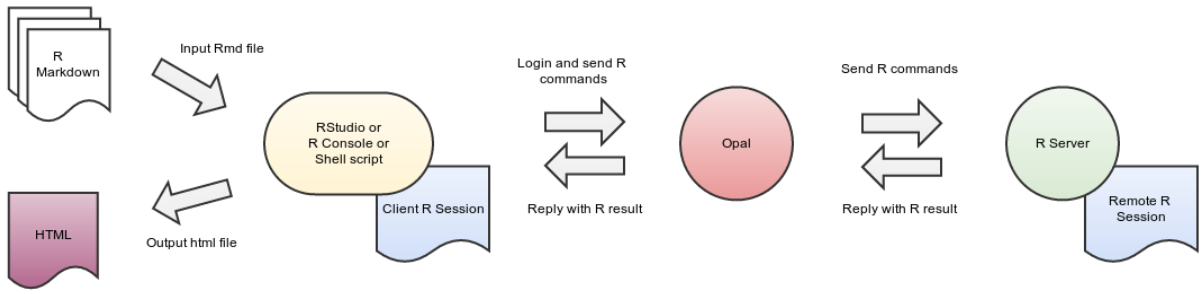
The result of this report is a document that presents the data dictionary of a Opal table with figures and summary statistics.

See Opal Data Dictionary with R output.

13.4 Report Execution Flows

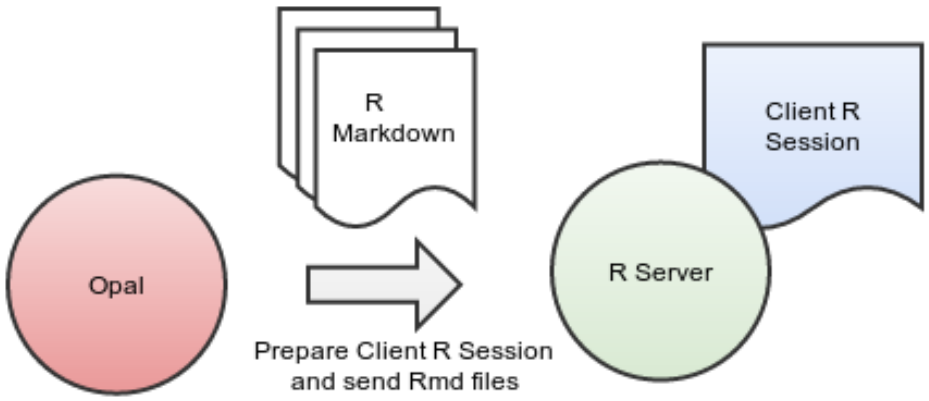
13.4.1 Local R Session

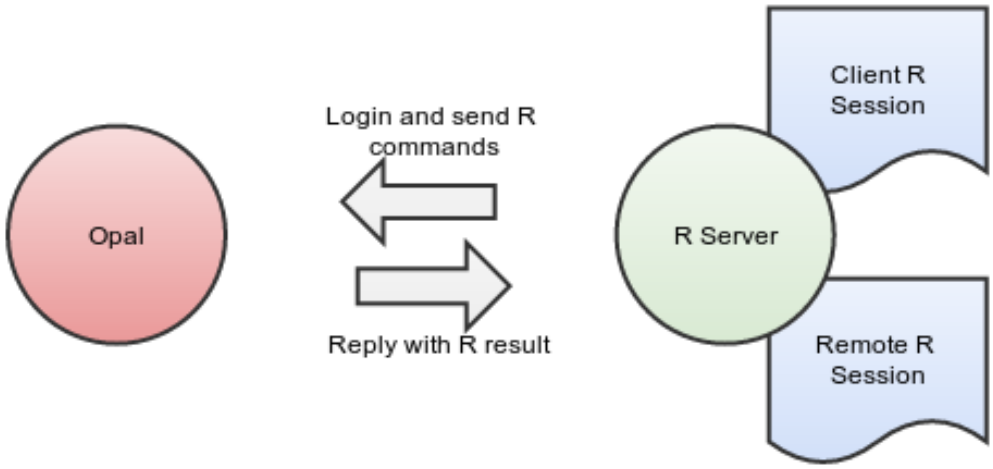
When executed in the context of RStudio, R console or Shell script, the communication flow is:

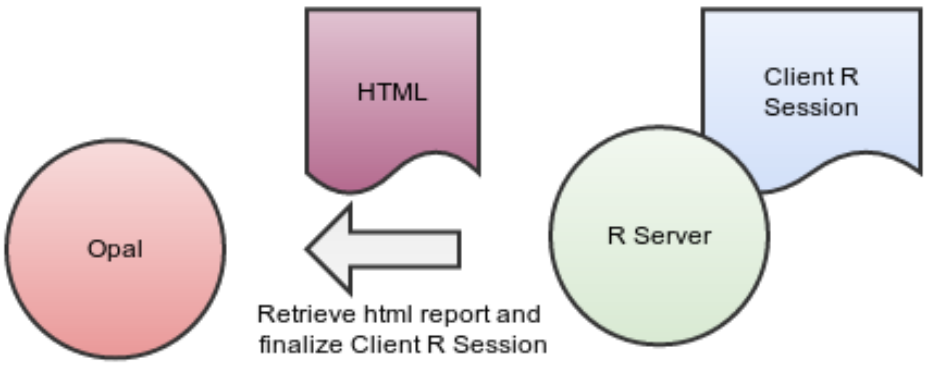


13.4.2 Opal R Session

When executed by Opal the communication flow is:


R
Markdown
Client R
Session
Opal
R Server
Prepare Client R Session
and send Rmd files


Client R
Session
Login and send R
commands
Opal
R Server
Reply with R result
Remote R
Session


HTML
Client R
Session
Opal
R Server
Retrieve html report and
finalize Client R Session

CHAPTER 14

Python Introduction

Opal Python client, a command line scripting tool written in Python, enables automation of tasks in a Opal server.

14.1 Requirements

Python 2.x must be installed on the system. See more about Python.

14.2 Installation

You can install Opal Python Client via the following two methods:

- use the Debian/RPM package manager
- use a Python package

14.2.1 Debian Package Installation

Follow the OBiBa Debian Repository instructions and run:

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

14.2.2 RPM Package Installation

Follow the OBiBa RPM Repository instructions and run:

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

14.2.3 Python Package Installation

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

Install on Linux or Mac

1. Download the most recent version
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
```

Note: 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 https://github.com/obiba/opal/releases/download/X.XX/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 Google 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 most recent version and follow these steps:

```
unzip opal-python-client-X.XX.tar.gz
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)

14.3 Usage

To get the options of the command line:

```
opal --help
```

This command will display which sub-commands are available. Further, given a subcommand obtained from command above, its help message can be displayed via:

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

This command will display available subcommands.

CHAPTER 15

Python Commands

15.1 Datasources Commands

These commands allow to access to both variables and values.

15.1.1 Data Dictionary

Get metadata: datasources, tables or variables.

```
opal dict <RESOURCE> <CREDENTIALS> [EXTRAS]
```

Arguments

Argument	Description
RESOURCE	Resource identification in the format <datasource>[.<table>[:<variable>]]. Wild card * is supported. Each project has a datasource which is identified by the project's name.

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

To fetch the dictionary associated to a datasource:

```
opal dict datashield --opal https://opal-demo.obiba.org --user administrator --
↪password password
```

To fetch the dictionary associated to a table in a pretty format:

```
opal dict datashield.CNSIM1 --opal https://opal-demo.obiba.org --user administrator --
↪password password -j
```

To fetch the description of a variable:

```
opal dict datashield.CNSIM1: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 datashield datasource
opal dict "datashield.*" --opal https://opal-demo.obiba.org --user administrator --
↪password password

# Get all variables datashield.CNSIM1 table
opal dict "datashield.CNSIM1:*" --opal https://opal-demo.obiba.org --user␣
↪administrator --password password
```

15.1.2 Data

Get data: list of entity identifiers or entity values from a table/variable.

```
opal data <RESOURCE> <CREDENTIALS> [EXTRAS]
```

Arguments

Argument	Description
RESOURCE	Resource identification in the format <datasource>.<table>[:<variable>]. Each project has a datasource which is identified by the project's name.

Credentials

Authentication is done by username/password credentials.

Option	Description
--opal OPAL, -o OPAL	Opal server base url.
--user USER, -u USER	User name. User with appropriate permissions is expected depending of the REST resource requested.
--password PASSWORD, -p PASSWORD	User 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

Option	Description
-h, --help	Show the command help's message.
--verbose, -v	Verbose output.
--json, -j	Pretty JSON formatting of the response.

Example

Fetch the list of entity identifiers associated to a table:

```
opal data datashield.CNSIM1 --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 all participant values in a table
opal data datashield.CNSIM1 -o https://opal-demo.obiba.org -u administrator -p␣
↪password --id 9553295965 -j
```

(continues on next page)

(continued from previous page)

```
# Get the JSON representation of a variable value
opal data datashield.CNSIM1:GENDER -o https://opal-demo.obiba.org -u administrator -p␣
↪password --id 9553295965 -j

# Get the raw value. If variable is of binary type, a byte stream is outputed.
opal data datashield.CNSIM1:GENDER -o https://opal-demo.obiba.org -u administrator -p␣
↪password --id 9553295965 -j --raw
```

15.1.3 Entity

Get entity information.

```
opal entity <ID> <CREDENTIALS> [EXTRAS]
```

Arguments

Argument	Description
ID	Entity identifier.

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.

Credentials

Authentication is done by username/password credentials.

Option	Description
--opal OPAL, -o OPAL	Opal server base url.
--user USER, -u USER	User name. User with appropriate permissions is expected depending of the REST resource requested.
--password PASSWORD, -p PASSWORD	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

Fetch the entities with id 9553295965:

```
opal entity 9553295965 --opal https://opal-demo.obiba.org --user administrator --
↪password password
```

Fetch the list of table where entity 9553295965 exists:

```
opal entity 9553295965 --opal https://opal-demo.obiba.org --user administrator --
↪password password --tables
```

Fetch the list of table where entity 9553295965 of type "Participant" exists:

```
opal entity 9553295965 --opal https://opal-demo.obiba.org --user administrator --
↪password password --tables --type Participant
```

15.1.4 Table Copy

Launch a task that will perform a table copy from one project to another. The destination project can be the one of origin in which case the table has to be renamed.

```
opal copy-table <CREDENTIALS> [OPTIONS] [XTRAS]
```

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

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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 datashield --tables CNSIM1 --destination datashield --name CNSIM4
```

15.1.5 Table Delete

Delete one or more tables of a project.

```
opal delete-table <CREDENTIALS> [OPTIONS] [XTRAS]
```

Options

Option	Description
`--project, -pr`	Source project name
`--tables, -t`	List of table names which will be deleted (default is all)

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.

Example

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
```

15.1.6 Annotations Import

Import the variable annotations of one or more tables. This can be used to restore annotations that were backed up using *Annotations Export*.

```
opal import-annot <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Options

Option	Description
`--input INPUT, -in INPUT`	CSV/TSV input file, typically the output of the export-annot command (default is stdin)
`--locale LOCALE, -l LOCALE`	Destination annotation locale (default is none)
`--separator SEPARATOR, -s SEPARATOR`	Separator char for CSV/TSV format (default is the tabulation character)
`--destination DESTINATION, -d DESTINATION`	Destination project 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)

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.

Example

Import some annotations to a specified table:

```
opal import-annot --user administrator --password password --destination Study2 --
↪tables datasetA --input /tmp/area-annotations.tsv
```

15.1.7 Annotations Export

Export the variable annotations of one or all tables of a project. This can be used to backup annotations, that can be restored using *Annotations Import*.

```
opal export-annot <RESOURCE> <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Arguments

Argument	Description
RESOURCE	Resource identification in the format <datasource>[.<table>]. Each project has a datasource which is identified by the project's name.

Options

Option	Description
--output OUTPUT, -out OUTPUT	CSV/TSV file path where to write the exported annotations. When not specified, standard output is used.
--locale LOCALE, -l LOCALE	Exported locale (default is none)
--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).

Credentials

Authentication is done by username/password credentials.

Option	Description
--opal OPAL, -o OPAL	Opal server base url.
--user USER, -u USER	User name. User with appropriate permissions is expected depending of the REST resource requested.
--password PASSWORD, -p PASSWORD	User 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

Option	Description
-h, --help	Show the command help's message.
--verbose, -v	Verbose output.

Example

Export annotations of the Mlstr_area taxonomy from a specific table:

```
opal export-annot --opal https://opal-demo.obiba.org --user administrator --password␣
↪password CLSA --taxonomies Mlstr_area --out /tmp/clsa-area.tsv
```

15.1.8 Backup Views

Backup named or all views of a project in the local file system. If the backup directory does not exist, it will be created. The views are stored as JSON files. Permissions that may have been setup are not backed up.

```
opal backup-view <CREDENTIALS> [OPTIONS] [XTRAS]
```

Options

Option	Description
`--project PROJECT, -pr PROJECT`	Source project name
`--views VIEWS [VIEWS ...], -vw VIEWS [VIEWS ...]`	List of view names to be backed up (default is all)
`--output OUTPUT, -out OUTPUT`	Output directory name (default is current directory)
`--force, -f`	Skip confirmation when overwriting the backup file.

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.

Example

Backup a specific view from a project into file `CNSIM.json`:

```
opal backup-view --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --project datashield --views CNSIM
```

15.1.9 Restore Views

Restore views that were previously backed up in the local file system into a project. The expected format of the view files is JSON. If one or more tables that are referenced by the backed up view do not exist anymore, the restoration will fail.

```
opal restore-view <CREDENTIALS> [OPTIONS] [XTRAS]
```

Options

Option	Description
`--project PROJECT, -pr PROJECT`	Destination project name
`--views VIEWS [VIEWS ...], -vw VIEWS [VIEWS ...]`	List of view names to be restored (default is all the JSON files that are found in the backup directory)
`--input INPUT, -in INPUT`	Input directory name (default is current directory)
`--force, -f`	Skip confirmation when overwriting an existing view.

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.

Example

Restore a specific view file `CNSIM.json` into a project:

```
opal restore-view --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --project datashield --views CNSIM
```

15.2 Import Commands

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

15.2.1 Import CSV

Import a CSV file, to be found in Opal file system.

```
opal import-csv <CREDENTIALS> <OPTIONS> [EXTRAS]
```

Options

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.
`--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)
`--valueType VALUETYPE, -vt VALUETYPE`	Default value type (text, integer, decimal, boolean etc.). When not specified, "text" is the default.

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.2.2 Import Opal Archive

Import an archive of XML files, to be found in Opal file system.

```
opal import-spss <CREDENTIALS> <OPTIONS> [EXTRAS]
```

Options

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.
`--path PATH -pa PATH`	Path to the zip of XML files to import on the Opal file system

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.2.3 Import SPSS

Import a SPSS file, to be found in Opal file system.

```
opal import-xml <CREDENTIALS> <OPTIONS> [EXTRAS]
```

Options

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.
`--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.

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.2.4 Import SAS (R)

Import a SAS or SAS Transport file, to be found in Opal file system, using R.

```
opal import-r-sas <CREDENTIALS> <OPTIONS> [EXTRAS]
```

Options

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.
`--path PATH -pa PATH`	Path to the SAS or SAS Transport file to import on the Opal file system
`--locale LOCALE, -l LOCALE`	Language code to be associated to 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.

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.2.5 Import SPSS (R)

Import a SPSS or compressed SPSS file, to be found in Opal file system, using R.

```
opal import-r-spss <CREDENTIALS> <OPTIONS> [EXTRAS]
```

Options

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.
`--path PATH -pa PATH`	Path to the SPSS or compressed SPSS file to import on the Opal file system
`--locale LOCALE, -l LOCALE`	Language code to be associated to 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.

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.2.6 Import Stata (R)

Import a Stata file, to be found in Opal file system, using R.

```
opal import-r-stata <CREDENTIALS> <OPTIONS> [EXTRAS]
```

Options

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.
`--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 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.

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.2.7 Import Opal

Import from a remote opal server.

```
opal import-opal <CREDENTIALS> <OPTIONS> [EXTRAS]
```

Options

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.
`--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
`--rdatasource RDATASOURCE, -rd RDATASOURCE`	Remote datasource name

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.2.8 Import LimeSurvey

Import from a remote LimeSurvey server.

```
opal import-limesurvey <CREDENTIALS> <OPTIONS> [EXTRAS]
```

Options

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.
`--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)

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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)"
```

15.2.9 Import SQL

Import from a remote SQL server.

```
opal import-sql <CREDENTIALS> <OPTIONS> [EXTRAS]
```

Options

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.
`--database DATABASE, -db DATABASE`	Name of the SQL database as registered in Opal

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.2.10 Import System Identifiers

Import entity system identifiers.

```
opal import-ids <CREDENTIALS> <OPTIONS> [EXTRAS]
```

Options

Option	Description
`--type TYPE, -t TYPE`	Entity type of the data (e.g. Participant)

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.2.11 Import Identifiers Mapping

Import Opal server entity identifiers.

```
opal import-ids-map <CREDENTIALS> <OPTIONS> [EXTRAS]
```

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 ",")

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.2.12 Import VCF

Import VCF file(s) from Opal file system. Requires that the destination project has a VCF store activated.

```
opal import-vcf <CREDENTIALS> <OPTIONS> [EXTRAS]
```

Options

Option	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)

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.3 Export Commands

Export one or more tables to the Opal file system.

15.3.1 Export CSV

Export in CSV format in Opal file system.

```
opal export-csv <CREDENTIALS> <OPTIONS> [EXTRAS]
```

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

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

Export tables from opal-data. A subdirectory 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
```

15.3.2 Export Opal Archive

Export in XML format in Opal file system.

```
opal export-xml <CREDENTIALS> <OPTIONS> [EXTRAS]
```

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 zip file name on the Opal file system

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

Export tables from opal-data to a ZIP file of XML files:

```
opal export-xml --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --datasource opal-data --tables Spirometry StandingHeight --output /tmp/
↪export.zip
```

15.3.3 Export SAS (R)

Export in SAS or SAS Transport format in Opal file system.

```
opal export-r-sas <CREDENTIALS> <OPTIONS> [EXTRAS]
```

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 SAS or SAS Transport file name on the Opal file system

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.3.4 Export SPSS (R)

Export in SPSS or compressed SPSS format in Opal file system.

```
opal export-r-spss <CREDENTIALS> <OPTIONS> [EXTRAS]
```

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 SPSS or compressed SPSS file name on the Opal file system

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.3.5 Export Stata (R)

Export in Stata format in Opal file system.

```
opal export-r-stata <CREDENTIALS> <OPTIONS> [EXTRAS]
```

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 Stata file name on the Opal file system

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.3.6 Export SQL

Export in a SQL database.

```
opal export-sql <CREDENTIALS> <OPTIONS> [EXTRAS]
```

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

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

Export tables from opal-data to a SQL database:

```
opal export-sql --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --datasource opal-data --tables Spirometry  --database sql_db
```

15.3.7 Export VCF

Export VCF files in Opal file system.

```
opal export-vcf <CREDENTIALS> <OPTIONS> [EXTRAS]
```

Options

Option	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)

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.4 User and Group Commands

Opal (internal) users and groups management. Does not apply to any Agate users and groups.

15.4.1 User

Manage a user in the Opal internal user directory.

```
opal user <CREDENTIALS> [OPTIONS] [EXTRAS]
```

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 PWD, -upa PWD`	User password of at least six characters.
`--ucertificate CERT, -uc CERT`	User certificate (public key) file
`--disabled, -di`	Disable user account (if omitted the user is enabled by default).
`--groups GRP [GRP ...], -g GRP [GRP ...]`	User groups (separated by space)

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Output pretty-print JSON

Example

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
```

15.4.2 Group

Manage a group in the Opal internal user directory.

```
opal group <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Options

Option	Description
`--name NAME, -n NAME`	Group name
`--fetch, -f`	Fetch a user (all groups are returned if no option –name is given)
`--delete, -de`	Delete the group specified by the –name option

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Output pretty-print JSON

Example

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
```

15.5 Permission Commands

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

15.5.1 Project Permission

Manage global permissions on the project.

```
opal perm-project <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Options

Option	Description
`--add, -a`	Add a permission
`--delete, -d`	Delete a permission
`--permission PERM, -pe PERM`	Permission to apply: `administrate`
`--subject SUBJECT, -s SUBJECT`	Subject name to which the permission will be granted
`--type TYPE, -ty TYPE`	Subject type: `user` or `group`
`--project PROJECT, -pr PROJECT`	Project name on which the permission is to be set

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Output pretty-print JSON

Example

Add administrate permission for subject demouser on datashield project:

```
opal perm-project --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --type USER --subject demouser --permission administrate --project␣
↪datashield --add
```

Remove the above permission:

```
opal perm-project --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --type USER --subject demouser --project datashield --delete
```

15.5.2 Datasource Permission

Manage permissions on the project's datasource (the set of tables).

```
opal perm-datasource <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Options

Option	Description
`--add, -a`	Add a permission
`--delete, -d`	Delete a permission
`--permission PERM, -pe PERM`	Permission to apply: `administrate`, `add-table`, `view-value`
`--subject SUBJECT, -s SUBJECT`	Subject name to which the permission will be granted
`--type TYPE, -ty TYPE`	Subject type: `user` or `group`
`--project PROJECT, -pr PROJECT`	Project name on which the permission is to be set

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Output pretty-print JSON

Example

Add add-table permission for subject demouser on datashield project:

```
opal perm-datasource --opal https://opal-demo.obiba.org --user administrator --
↪password password --type USER --subject demouser --permission add-table --project␣
↪datashield --add
```

Remove the above permission:

```
opal perm-datasource --opal https://opal-demo.obiba.org --user administrator --
↪password password --type USER --subject demouser  --project datashield --delete
```

15.5.3 Table Permission

Manage permissions on a project's table.

```
opal perm-table <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Options

Option	Description
`--add, -a`	Add a permission
`--delete, -d`	Delete a permission
`--permission PERM, -pe PERM`	Permission to apply: `view`, `view-value`, `edit`, `edit-values`, `administrate`
`--subject SUBJECT, -s SUBJECT`	Subject name to which the permission will be granted
`--type TYPE, -ty TYPE`	Subject type: `user` or `group`
`--project PROJECT, -pr PROJECT`	Project name on which the permission is to be set
`--tables TABLE [TABLE ...], -t TABLE [TABLE ...]`	List of table names on which the permission is to be set (default is all)

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Output pretty-print JSON

Example

Add view permission for subject demouser on table CNSIM1 in datashield project:

```
opal perm-table --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --type USER --project datashield --subject demouser  --permission view --
↪add --tables CNSIM1
```

Remove the above permission:

```
opal perm-table --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --type USER --project datashield --subject demouser --delete --table CNSIM1
```

Add permission on all tables of datashield project:

```
opal perm-table --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --type USER --project datashield --subject demouser --permission view --add
```

Remove permission from all table of datashield project:

```
opal perm-table --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --type USER --project datashield --subject demouser --delete
```

Add permission on specific tables:

```
opal perm-table --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --type USER --project datashield --subject demouser --permission view --
↪add --tables CNSIM1 FNAC
```

15.5.4 Variable Permission

Manage permissions on a project's variable (in a table).

```
opal perm-variable <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Options

Option	Description
`--add, -a`	Add a permission
`--delete, -d`	Delete a permission
`--permission PERM, -pe PERM`	Permission to apply: `view`
`--subject SUBJECT, -s SUBJECT`	Subject name to which the permission will be granted
`--type TYPE, -ty TYPE`	Subject type: `user` or `group`
`--project PROJECT, -pr PROJECT`	Project name on which the permission is to be set
`--table TABLE, -t TABLE`	Table name to which the variables belong
`--variables VARIABLE [VARIABLE ...], -va VARIABLE [VARIABLE ...]`	List of variable names on which the permission is to be set

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Output pretty-print JSON

Example

Add view permission for subject demouser on variables GENDER and LAB_HDL of table CNSIM1 in datashield project:

```
opal perm-variable --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --type USER --subject demouser --project datashield --table CNSIM1 --
↪variables GENDER LAB_HDL --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 datashield --table CNSIM1 --
↪variables GENDER LAB_HDL --delete
```

15.5.5 System Permission

Manage global system permissions.

```
opal perm-system <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Options

Option	Description
`--add, -a`	Add a permission
`--delete, -d`	Delete a permission
`--permission PERM, -pe PERM`	Permission to apply: `administrate`, `add-project`
`--subject SUBJECT, -s SUBJECT`	Subject name to which the permission will be granted
`--type TYPE, -ty TYPE`	Subject type: `user` or `group`

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Output pretty-print JSON

Example

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
```

15.5.6 R Permission

Manage R server usage permissions.

```
opal perm-r <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Options

Option	Description
`--add, -a`	Add a permission
`--delete, -d`	Delete a permission
`--permission PERM, -pe PERM`	Permission to apply: `use`
`--subject SUBJECT, -s SUBJECT`	Subject name to which the permission will be granted
`--type TYPE, -ty TYPE`	Subject type: `user` or `group`

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Output pretty-print JSON

Example

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 --delete
```

15.5.7 DataSHIELD Permission

Manage DataSHIELD usage permissions.

```
opal perm-datashield <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Options

Option	Description
`--add, -a`	Add a permission
`--delete, -d`	Delete a permission
`--permission PERM, -pe PERM`	Permission to apply: `administrate`, `use`
`--subject SUBJECT, -s SUBJECT`	Subject name to which the permission will be granted
`--type TYPE, -ty TYPE`	Subject type: `user` or `group`

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Output pretty-print JSON

Example

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
```

15.6 Other Commands

Other commands, for advanced users.

15.6.1 File

Manage files in the Opal file system.

```
opal file <PATH> <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Arguments

Argument	Description
PATH	The path of the file in the Opal file system

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

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.6.2 System

Query for system status and configuration.

```
opal system <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Options

Option	Description
`--conf`	Opal application configuration (default option)
`--version`	Opal version number
`--status`	Opal application status (JVM related dynamic properties)

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

15.6.3 Plugin

Manage system plugins.

```
opal plugin <CREDENTIALS> [OPTIONS] [EXTRAS]
```

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 INSTALL, -i INSTALL`	Install a plugin by providing its name or name:version or a path to a plugin archive file (in Opal file system). If no version is specified, the latest version is installed. Requires system restart to be effective.
`--remove REMOVE, -rm REMOVE`	Remove a plugin by providing its name. Requires system restart to be effective.
`--reinstate REINSTATE, -ri REINSTATE`	Reinstate a plugin that was previously removed by providing its name.
`--fetch FETCH, -f FETCH`	Get the named plugin description.
`--configure CONFIGURE, -c CONFIGURE`	Configure the plugin site properties. Usually requires to restart the associated service to be effective.
`--status STATUS, -su STATUS`	Get the status of the service associated to the named plugin.
`--start START, -sa START`	Start the service associated to the named plugin.
`--stop STOP, -so STOP`	Stop the service associated to the named plugin.

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

List the plugins that are currently installed:

```
opal plugin -o https://opal-demo.obiba.org -u administrator -p password --list --json
```

15.6.4 Task

Manage a task, for shell scripts submitting tasks (import/export) and waiting for their completion.

```
opal task <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Options

Option	Description
`--id ID`	The task ID. If not provided, it will be read from the standard input (from the JSON representation of the task or a plain value).
`--show, -sh`	Show JSON representation of the task
`--status, -st`	Get the status of the task
`--wait, -w`	Wait for the task to complete (successfully or not)
`--cancel, -c`	Cancel the task
`--delete, -d`	Delete the task

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

Get the full status of a task:

```
opal task -o https://opal-demo.obiba.org -u administrator -p password --id 1 --json
```

Get the status token of a task:

```
opal task -o https://opal-demo.obiba.org -u administrator -p password --id 1 --status
```

Wait for the task to complete and get its status token:

```
opal task -o https://opal-demo.obiba.org -u administrator -p password --id 1 --wait
```

15.6.5 Web Services

This command is for advanced users wanting to directly access to the REST API of Opal server.

```
opal rest <PATH> <CREDENTIALS> [OPTIONS] [EXTRAS]
```

Arguments

Argument	Description
`PATH`	Web service path, for instance: /project/xxx

Options

Option	Description
`--method METHOD, -m METHOD`	HTTP method: GET (default), POST, PUT, DELETE, OPTIONS.
`--accept ACCEPT, -a ACCEPT`	Accept header (default is application/json).
`--content-type CONTENT_TYPE, -ct CONTENT_TYPE`	Content-Type header (default is application/json).
`--headers HEADERS, -hs HEADERS`	Custom headers in the form of: { "Key2": "Value2", "Key2": "Value2" }

Credentials

Authentication is done by username/password credentials.

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User 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

Option	Description
`-h, --help`	Show the command help's message.
`--verbose, -v`	Verbose output.
`--json, -j`	Pretty JSON formatting of the response.

Example

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
```

CHAPTER 16

Magma JS Introduction

The Magma Javascript API is there to allow accession of data and data dictionary in order to perform data cleansing, data harmonization, data filtering etc.

16.1 Syntax

16.1.1 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.

16.1.2 Chaining

Methods from this API return an object to which a method can be directly applied. This allow method calls chaining, for instance:

```
$('DO_YOU_SMOKE').any('DNK', 'PNA').not()
```

16.2 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:

16.2.1 if-else

```
if($('BooleanType.blood.contraindicated').value()) {
   log('Blood collection has been contraindicated');
}
if($('IntegerType.tubes.collected').gt($('IntegerType.tubes.expected')).value()) {
   log('More tubes than expected.');
} else if ($('IntegerType.tubes.collected').lt($('IntegerType.tubes.expected')).
↪value()) {
   log('Less tubes than expected.');
} else {
   log('Collected tubes matches expected tubes.');
}
```

16.2.2 switch

```
switch($('Admin.Participant.gender').value()) {
   case "MALE":
      log('Participant is male.');
   case "FEMALE":
      log('Participant is female.');
   default:
      log('Participant gender is unknown.');
}
```

16.3 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.

CHAPTER 17

Magma JS Methods

17.1 Global Methods

17.1.1 $, $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 multilines 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.

See also *$group*, *$id, $identifier*, *$join*, *$this*, *$var, $variable*.

Syntax

```
$(name)
// alternate syntax
$val(name)
$value(name)
```

Parameter	Description
name	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.

17.1.2 $this

The current object is a value set in a view. $this will access to a variable value within this value set.

See also *$, $val, $value*, *$group*, *$id, $identifier*, *$join*, *$var, $variable*

Syntax

```
$this(name)
```

Parameter	Description
name	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')
```

17.1.3 $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.

See also *$, $val, $value*.

Syntax

```
$join(name,idname[,flat])
```

Parameter	Description
`name`	The name of the variable from which the value shall be retrieved.
`idname`	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, a table with a repeatable variable named **code**:

ID	code
aa	Acode1,Acode2
bb	Acode1
cc	Acode3
dd	

and **code_mapper** a table with 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 sequences:

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	

17.1.4 $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.

See also *$, $val, $value*, *$id, $identifier*, *$join*, *$var, $variable*

Syntax

```
$group(name,criteria[,select])
```

Parameter	Description
`name`	The name of the variable on which the selection criteria is to be applied.
`criteria`	Value to be compared to or a function for evaluating the matching criteria.
`select`	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
```

(continues on next page)

(continued from previous page)

```
$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()
```

17.1.5 $id, $identifier

The current object is a value set. $id will access to the entity Opal identifier within this value set.

See also *$, $val, $value*, *$join*, *$var, $variable*.

Syntax

```
$id()
// alternate syntax
$identifier()
```

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'))
```

17.1.6 $var, $variable

Returns the variable object at the given name. The name resolution is done given an execution context.

See also *$, $val, $value*, *$id, $identifier*.

Syntax

```
$var(name)
// alternate syntax
$variable(name)
```

Parameter	Description
name	The name that identifies the variable.

Examples

Get the variable with name DO_YOU_SMOKE.

```
$var('DO_YOU_SMOKE')
```

17.1.7 log

Provides *info* level logging of messages and variable values.

Syntax

```
log(text[, value_i[, ...]])
```

Parameter	Description
text	The text to be logged. May contain place holders for values.
value_i	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'));
```

17.1.8 now

Returns the current date time wrapped in a value object.

Syntax

```
now()
```

Examples

Get the current date time.

```
now()
```

17.1.9 newValue

Creates a new value object.

See also *asSequence*, *newSequence*.

Syntax

```
newValue(data[,type])
```

Parameter	Description
`data`	The data to be wrapped in the value object.
`type`	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')
```

(continues on next page)

(continued from previous page)

```
// 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())
```

17.1.10 newSequence

Creates a new value sequence object. A value sequence is-a value object but also has-some value objects (see Value Sequence Methods).

See also *newValue*, *asSequence*.

Syntax

```
newSequence(data[,type])
```

Parameter	Description
`data`	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`	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'])
```

17.1.11 $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.

See also *$lastupdate*.

Syntax

```
$created()
```

Examples

Check if a value set was created after a given date time.

```
$created().after(newValue('2014-05-05 10:30', 'datetime'))
```

17.1.12 $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.

See also *$created*.

Syntax

```
$lastupdate()
```

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'))
```

17.1.13 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.

Syntax

```
source(path)
```

Parameter	Description
`path`	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 located in Opal file system at path **/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());
```

17.2 Variable Methods

17.2.1 attribute

Get the variable attribute value with the given name.

See also *$var, $variable*.

Syntax

```
attribute(name)
```

Parameter	Description
`name`	The name that identifies the attribute.

Examples

Get the value corresponding to the 'stage' attribute:

```
$var('AVG_SITTING_HEIGHT').attribute('stage')
```

17.2.2 name

Get the name of the variable as a Text value.

See also *$var, $variable*.

Syntax

```
name()
```

Examples

```
$('AVar').name()
```

17.2.3 type

Returns the *Value Types* of the variable, as a Text value.

See also *$var, $variable*.

Syntax

```
type()
```

Examples

```
$var('DO_YOU_SMOKE').type()
```

17.2.4 entityType

Returns the entity type of the variable, as a Text value.

See also *$var, $variable*.

Syntax

```
entityType()
```

Examples

```
$var('DO_YOU_SMOKE').entityType()
```

17.2.5 refEntityType

Returns the referenced entity type of the variable, as a Text value.

See also *$var, $variable*.

Syntax

```
refEntityType()
```

Examples

```
$var('SampleID').refEntityType()
```

17.2.6 repeatable

Returns if the variable is repeatable, as a Boolean value.

See also *$var, $variable*.

Syntax

```
repeatable()
```

Examples

```
$var('Measure').repeatable()
```

17.2.7 occurrenceGroup

Returns the occurrence group of the variable, as a Text value.

See also *$var, $variable*.

Syntax

```
occurrenceGroup()
```

Examples

```
$var('HEIGHT').occurrenceGroup()
```

17.2.8 mimeType

Returns the media type of the variable, as a Text value.

See also *$var, $variable*.

Syntax

```
mimeType()
```

Examples

```
$var('OutFile').mimeType()
```

17.2.9 unit

Returns the unit of the variable, as a Text value.

See also *$var, $variable*.

Syntax

```
unit()
```

Examples

```
$var('HEIGHT').unit()
```

17.2.10 nature

Returns the nature of the variable as a Text value. The nature indicates the kind of summary statistics that can be performed on values. Nature values are:

- `CATEGORICAL`: the values of the variable are discrete and described by the categories of the variable or the value type is `boolean`,
- `CONTINUOUS`: values can be any in the considered value type (applies also to variables having categories that are all flagged as being *missing*),
- `TEMPORAL`: value of type `date` or `datetime`,
- `GEO`: value of type `point`, `linestring` or `polygone`,
- `BINARY`: value of type `binary`,
- `UNDETERMINED` when non of the above apply.

See also *$var, $variable*.

Syntax

```
nature()
```

Examples

```
$('VAR1').nature()
```

17.2.11 isNumeric

Returns whether the variable value type is `integer` or `decimal`, as a Boolean value.

See also *$var, $variable*.

Syntax

```
isNumeric()
```

Examples

```
$var('WEIGHT').isNumeric()
```

17.2.12 isDateTime

Returns whether the variable value type is `date` or `datetime`, as a Boolean value.

See also *$var, $variable*.

Syntax

```
isDateTime()
```

Examples

```
$var('Measurement.start').isDateTime()
```

17.2.13 isGeo

Returns whether the variable value type is `point`, `linestring` or `polygon`, as a Boolean value.

See also *$var, $variable*.

Syntax

```
isGeo()
```

Examples

```
$var('PostalCode.area').isGeo()
```

17.3 Value Methods

17.3.1 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.

See also *$, $val, $value*.

Syntax

```
all(param_1[, param_i[, ...]])
```

Parameter	Description
`param_i`	A string or a Value to be compared to.

Examples

```
$('CategoricalVar').all('CAT1', 'CAT2')
```

17.3.2 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.

See also *$, $val, $value*.

Syntax

```
any(param_1[, param_i[, ...]])
```

Parameter	Description
`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()
```

17.3.3 isNull

Returns true if the value is null.

See also *$, $val, $value*.

Syntax

```
isNull()
```

Examples

```
$('MyVar').isNull()
```

17.3.4 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.

See also *$, $val, $value*.

Syntax

```
whenNull(someValue)
```

Parameter	Description
someValue	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.
```

(continues on next page)

(continued from previous page)

```
$('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);
```

17.3.5 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.

See also *$, $val, $value*.

Syntax

```
map({key_1:value_1[, key_i:value_i[, ...]]}[, defaultValue[, nullValue]])
map(mapper)
```

Parameter	Description
`key_i:value_i`	A list of key/value pairs: the value to be mapped and 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) })
```

17.3.6 not

Returns the contrary of a boolean value or return if it does not match any of the arguments.

See also *$, $val, $value*.

Syntax

```
not([value1[,value2[, ...]]])
```

Parameter	Description
value_i	Optional 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'))
```

17.3.7 type

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.

See also *$, $val, $value*.

Syntax

```
type([name])
```

Parameter	Description
`type`	Optional type name into which the value shall be converted. Possible values are (case insensitive): `text`, `integer`, `decimal`, `boolean`, `locale`, `datetime`, `date`, `binary`, `point`, `linestring`, `polygon` (see *Value Types* description).

Examples

```
```

17.3.8 value

Returns the primitive javascript value from the Value.

See also *$, $val, $value*.

Syntax

```
value()
```

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()
```

(continues on next page)

(continued from previous page)

```
// 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]
```

17.3.9 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.

See also *$, $val, $value*.

Syntax

```
length()
```

Examples

Get the binary value's length in bytes:

```
$('BINARY').length()
```

17.4 Value Sequence Methods

17.4.1 empty

Returns true value if is operating on a sequence that contains zero values. Otherwise false value is returned.

See also *size*.

Syntax

```
empty()
```

Examples

```
$('Admin.Interview.exportLog.destination').empty()
```

17.4.2 first

Returns the first value in a value sequence.

See also *firstNotNull*, *valueAt*, *last*.

Syntax

```
first()
```

Examples

```
$('Admin.StageInstance.stage').first()
```

17.4.3 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.

See also *empty*, *first*, *last*, *valueAt*, *size*.

Syntax

```
firstNotNull()
```

Examples

```
$('Admin.StageInstance.stage').firstNotNull()
```

17.4.4 indexOf

Returns the first position of a value in a value sequence.

See also *lastIndexOf*.

Syntax

```
indexOf(value)
```

Parameter	Description
`value`	A primitive value or a value object to be looked for.

Examples

```
$('Admin.StageInstance.stage').indexOf('Spirometry')
```

17.4.5 last

Returns the last value in a value sequence.

See also *valueAt*, *first*.

Syntax

```
last()
```

Examples

```
$('Admin.StageInstance.stage').last()
```

17.4.6 lastIndexOf

Returns the last position of a value in a value sequence.

See also *indexOf*.

Syntax

```
lastIndexOf(value)
```

Parameter	Description
`value`	A primitive value or a value object to be looked for.

Examples

```
$('Admin.StageInstance.stage').lastIndexOf('Spirometry')
```

17.4.7 valueAt

Returns the value at a specified index within the sequence (0-based).

See also *first*, *last*.

Syntax

```
valueAt(index)
```

Parameter	Description
index	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 *$, $val, $value* 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 specific 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')
```

17.4.8 size

Returns the number of values within a sequence. Returns null value if operand is a null value.

See also *empty*.

Syntax

```
size()
```

Examples

```
$('Admin.StageInstance.stage').size()
```

17.4.9 reduce

Reduce values of a sequence to a single value using a custom javascript accumulating function. The reduction result is the last accumulated value.

See also *map*.

Syntax

```
reduce(accumulator[,initialValue])
```

Parameter	Description
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)
```

17.4.10 filter

Filter values of a sequence using a custom javascript predicating function.

See also *subset*, *map*, *reduce*.

Syntax

```
filter(predicate)
```

Parameter	Description
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();
})
```

17.4.11 subset

Filter values of a sequence by subsetting the values according to their position.

See also *filter*.

Syntax

```
subset(from[,to])
```

Parameter	Description
`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;
})
```

17.4.12 trimmer

Filter values of a sequence by removing null values.

See also *filter*, *subset*.

Syntax

```
trimmer()
```

Examples

Trim sequence values and its equivalent filter:

```
$('SequenceVar').trimmer()

// is equivalent to the filter:
$('SequenceVar').filter(function(value) {
  return value.isNull().not();
})
```

17.4.13 sort

Sorts a sequence in natural order of its values or using a custom javascript comparing function.

See also *first*, *last*.

Syntax

```
sort([function])
```

Parameter	Description
`function`	Optional javascript comparing function.

Examples

```
$('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();
})
```

17.4.14 max

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.

See also *min*.

Syntax

```
max()
```

Examples

```
$('StandingHeight:Measure.RES_HEIGHT_MEASURE').max();
```

17.4.15 min

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.

See also *max*.

Syntax

```
min()
```

Examples

```
$('StandingHeight:Measure.RES_HEIGHT_MEASURE').min();
```

17.4.16 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.

See also *max*, *min*, *sum*, *stddev*, *reduce*.

Syntax

```
avg()
```

Examples

```
$('StandingHeight:Measure.RES_HEIGHT_MEASURE').avg();
```

17.4.17 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.

See also *max*, *min*, *stddev*, *avg*, *reduce*.

Syntax

```
sum()
```

Examples

```
$('StandingHeight:Measure.RES_HEIGHT_MEASURE').sum();
```

17.4.18 stddev

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.

See also *max*, *min*, *sum*, *avg*, *reduce*.

Syntax

```
stddev()
```

Examples

```
$('StandingHeight:Measure.RES_HEIGHT_MEASURE').stddev();
```

17.4.19 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.

See also *append*, *insertAt*, *prepend*.

Syntax

```
push(value [,value])
```

Parameter	Description
`value`	The value(s) 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_PULSE')).avg();
```

17.4.20 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.

See also *insertAt*, *prepend*, *push*.

Syntax

```
append(value [,value])
```

Parameter	Description
`value`	The value(s) 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);
```

17.4.21 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.

See also *append*, *insertAt*, *push*.

Syntax

```
prepend(value [,value])
```

Parameter	Description
`value`	The value(s) 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_PULSE')).
↪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);
```

17.4.22 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.

See also *append*, *prepend*, *push*.

Syntax

```
insertAt(position, value [,value])
```

Parameter	Description
`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(s) to insert in the sequence

Examples

```
// Add a value to a sequence, then compute the average of the resulting sequence
$('BloodPressure:Measure.RES_PULSE').insertAt(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);
```

17.4.23 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.

See also *zip*.

Syntax

```
join([delimiter[, prefix[, suffix]]])
```

Parameter	Description
`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();
```

17.4.24 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.

See also *join*.

Syntax

```
zip(value_1[, value_2[, ...]]], function)
```

Parameter	Description
value	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);
  });
```

17.4.25 asSequence

Turns a value object into a value sequence object. A value sequence *is-a* value object but also *has-some* value objects. If the value object on which it is applied is already a value sequence object, no operation is made.

See also *newValue*, *newSequence*, *isSequence*.

Syntax

```
asSequence()
```

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()
```

17.4.26 isSequence

Check whether a value is a value sequence object.

See also *asSequence*.

Syntax

```
isSequence()
```

Examples

Would be True if the VAR is a repeatable variable or if VAR is in different tables referred by the view:

```
$('VAR').isSequence()
```

17.5 Boolean Value Methods

17.5.1 and

Applies the ternary AND logic on values. If no arguments is provided, returns the value of the left operand.

See also *or*.

Syntax

```
and(value_1[, value_i[, ...]])
```

Parameter	Description
value_i	The boolean value(s) to be compared to.

Examples

```
$('BooleanVar').and($('OtherBooleanVar'))
$('BooleanVar').and($('OtherBooleanVar').not())
$('BooleanVar').and($('SomeBooleanVar'), $('OtherBooleanVar'))
```

17.5.2 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.

See also *eq*.

Syntax

```
compare(value)
```

Parameter	Description
value	The value to be compared to.

Examples

```
$('AVar').compare($('OtherVar'));
```

17.5.3 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.

See also *compare*.

Syntax

```
eq(value_1[, value_i[, ...]])
```

Parameter	Description
value_i	The value to be compared to, can be a primitive type or a Value object.

Examples

```
$('AVar').eq($('OtherVar'));
```

Check sequence equality:

```
$('VAR1').eq('a','b');
```

17.5.4 or

Applies the ternary OR logic on values. If no arguments is provided, returns the value of the left operand.

See also *and*.

Syntax

```
or(value_1[, value_i[, ...]])
```

Parameter	Description
value_i	The boolean value(s) to be compared to.

Examples

```
$('BooleanVar').or($('OtherBooleanVar'))
$('BooleanVar').or($('OtherBooleanVar').not())
$('BooleanVar').or($('SomeBooleanVar'), $('OtherBooleanVar'))
```

17.6 Numeric Value Methods

17.6.1 ge

Returns if left operand value is greater equal than right operand value. The operands must be either be of integer or decimal type.

See also *gt*, *le*, *lt*.

Syntax

```
ge(value)
```

Parameter	Description
value	A primitive value or a value object to be compared to.

Examples

```
$('AVar').ge($('OtherVar'))
$('HEIGHT').ge(100)
```

17.6.2 gt

Returns if left operand value is greater than right operand value. The operands must be either be of integer or decimal type.

See also *ge*, *le*, *lt*.

Syntax

```
gt(value)
```

Parameter	Description
value	A primitive value or a value object to be compared to.

Examples

```
$('AVar').gt($('OtherVar'))
$('HEIGHT').gt(100)
```

17.6.3 le

Returns if left operand value is lower equal than right operand value. The operands must be either be of integer or decimal type.

See also *ge*, *gt*, *lt*.

Syntax

```
le(value)
```

Parameter	Description
value	A primitive value or a value object to be compared to.

Examples

```
$('AVar').le($('OtherVar'))
$('HEIGHT').le(100)
```

17.6.4 lt

Returns if left operand value is lower than right operand value. The operands must be either be of integer or decimal type.

See also *ge*, *gt*, *le*.

Syntax

```
lt(value)
```

Parameter	Description
`value`	A primitive value or a value object to be compared to.

Examples

```
$('AVar').lt($('OtherVar'))
$('HEIGHT').lt(100)
```

17.6.5 plus

Returns result of first operand value plus second operand value. The operands must be either of integer or decimal type.

See also *minus*, *multiply*, *div*.

Syntax

```
plus(value)
```

Parameter	Description
`value`	A primitive value or a value object.

Examples

```
$('AVar').plus($('OtherVar'))
$('HEIGHT').plus(100)
```

17.6.6 minus

Returns result of first operand value minus second operand value. The operands must be either of integer or decimal type.

See also *plus*, *multiply*, *div*.

Syntax

```
minus(value)
```

Parameter	Description
value	A primitive value or a value object.

Examples

```
$('AVar').minus($('OtherVar'))
$('HEIGHT').minus(100)
```

17.6.7 multiply

Returns result of first operand value multiply second operand value. The operands must be either of integer or decimal type.

See also *plus*, *minus*, *div*.

Syntax

```
multiply(value)
```

Parameter	Description
value	A primitive value or a value object.

Examples

```
$('AVar').multiply($('OtherVar'))
$('HEIGHT').multiply(100)
```

17.6.8 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.

See also *plus*, *minus*, *multiply*.

Syntax

```
div(value)
```

Parameter	Description
`value`	A primitive value or a value object.

Examples

```
$('AVar').div($('OtherVar'))
$('HEIGHT').div(100)
```

17.6.9 ln

Returns the natural logarithm (base e) of the value. To obtain other logarithms, use the *log* method.

See also *log*.

Syntax

```
ln()
```

Examples

```
$('AVar').ln()
```

17.6.10 log

Returns the base 10 (or a specific base) logarithm of the value.

See also *ln*.

Syntax

```
log([base])
```

Parameter	Description
`base`	The optional base of the logarithm to evaluate (default is `10`).

Examples

```
$('AVar').log()
$('AVar').log(2) // base-2 logarithm
```

17.6.11 abs

Returns the absolute value of the current value.

Syntax

```
abs()
```

Examples

```
$('AVar').abs()
```

17.6.12 pow

Returns the value raised to the power of the argument.

Syntax

```
pow([power])
```

Parameter	Description
power	The power to raise the value to.

Examples

```
$('AVar').pow(2) // AVar * AVar
$('AVar').pow(-1) // 1 / AVar
```

17.6.13 sqroot

Returns the square root of the value.

See also *root*, *cbroot*.

Syntax

```
sqroot()
```

Examples

```
$('AVar').sqroot()
```

17.6.14 cbroot

Returns the cubic root of the value.

See also *root*, *sqroot*.

Syntax

```
cbroot()
```

Examples

```
$('AVar').cbroot()
```

17.6.15 root

Returns the root of the value.

See also *cbroot*, *sqroot*.

Syntax

```
root(root)
```

Parameter	Description
root	The root to evaluate.

Examples

```
$('AVar').root(2) // same as sqroot()
$('AVar').root(3) // same as cbroot()
```

17.6.16 round

Returns the rounded value of the current value.

Syntax

```
round([scale])
```

Parameter	Description
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)
```

17.6.17 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.

See also *map*.

Syntax

```
group(array_of_bounds[, array_of_outliers])
```

Parameter	Description
`array_of_bounds`	A list of values that will be the bounds of the ranges.
`array_of_outliers`	An optional 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
 });
```

17.7 Text Value Methods

17.7.1 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.

See also *compare*.

Syntax

```
compareNoCase(value)
```

Parameter	Description
value	The value to be compared to, ignoring case.

Examples

```
$('AVar').compareNoCase($('OtherVar'));
```

17.7.2 concat

Returns the text type result of first operand concat second operand. The operands must be either values or text type.

See also *capitalize*, *lowerCase*, *replace*, *trim*, *upperCase*.

Syntax

```
concat(value_1[, value_i[, ...]])
```

Parameter	Description
value_i	The value which string representation is to be concatenated to the text value.

Examples

```
$('AVar').concat($('OtherVar'));
```

17.7.3 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

See also *datetime*.

Syntax

```
date(format)
```

Parameter	Description
`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")
```

17.7.4 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

See also *date*.

Syntax

```
datetime(format)
```

Parameter	Description
`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').datetime('MM/dd/yy h:mm a')

// example: Wed, 4 Jul 2001 12:08:56 -0700
$('AVar').datetime('EEE, d MMM yyyy HH:mm:ss Z')

// example: 2008-03-01T13:00:00+01:00
$('AVar').datetime("yyyy-MM-dd'T'HH:mm:ssZ")
```

17.7.5 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.

See also *replace*.

Syntax

```
matches(regex)
```

Parameter	Description
`regex`	The regular expression to be searched for.

Examples

```
$('VarName').matches(/yes/)
```

17.7.6 replace

Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.

See also *matches*.

Syntax

```
replace(regex, text)
```

Parameter	Description
`regex`	The regular expression to be searched for.
`text`	The string that replaces the matched substring.

Examples

```
$('VarName').replace(/yes/, '1')
```

17.7.7 trim

Returns a copy of the string, with leading and trailing whitespace omitted.

See also *capitalize*, *lowerCase*, *replace*, *concat*, *upperCase*.

Syntax

```
trim()
```

Examples

```
$('AVar').trim();
```

17.7.8 lowerCase

Returns a copy of the string, lower cased.

See also *capitalize*, *concat*, *replace*, *trim*, *upperCase*.

Syntax

```
lowerCase([locale])
```

Parameter	Description
`locale`	The optional locale to be used, as a string with syntax: `language[_country[_variant]]`.

Examples

```
$('AVar').lowerCase()
$('AVar').lowerCase('fr_CA')
```

17.7.9 upperCase

Returns a copy of the string, upper cased.

See also *capitalize*, *lowerCase*, *replace*, *trim*, *concat*.

Syntax

```
upperCase([locale])
```

Parameter	Description
`locale`	The optional locale to be used, as a string with syntax: `language[_country[_variant]]`.

Examples

```
$('AVar').upperCase()
$('AVar').upperCase('fr_CA')
```

17.7.10 capitalize

Returns a copy of the string, with first characters of each word capitalized.

See also *upperCase*, *lowerCase*, *replace*, *trim*, *concat*.

Syntax

```
capitalize([delimiters])
```

Parameter	Description
delimiters	The optional delimiting characters for identifying words. Default is ' '.

Examples

```
$('AVar').capitalize()
$('AVar').capitalize(':;,( .["')
```

17.8 Date and Datetime Value Methods

17.8.1 add

Adds days to a value of date or date time type.

Syntax

```
add(days)
```

Parameter	Description
days	The number of days to be added.

Examples

Adds 2 days:

```
$('Date').add(2)
```

Subtracts 4 days:

```
$('Date').add(-4)
```

17.8.2 after

Returns true if the date value is after the specified date value(s).

See also *before*.

Syntax

```
after(date_1[, date_2[, ...]])
```

Parameter	Description
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'))
```

17.8.3 before

Returns true if the date value is before the specified date value(s).

See also *after*.

Syntax

```
before(date_1[, date_2[, ...]])
```

Parameter	Description
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'))
```

17.8.4 dayOfMonth

Returns the day of month from a date as an integer starting from 1.

See also *dayOfWeek*, *dayOfYear*.

Syntax

```
dayOfMonth()
```

Examples

```
$('Date').dayOfMonth()
```

17.8.5 dayOfWeek

Returns the day of week from a date as an integer starting from 1 (Sunday).

See also *dayOfMonth*, *dayOfYear*.

Syntax

```
dayOfWeek()
```

Examples

```
$('Date').dayOfWeek()
```

17.8.6 dayOfYear

Returns the day of year from a date as an integer starting from 1.

See also *dayOfMonth*, *dayOfWeek*.

Syntax

```
dayOfYear()
```

Examples

```
$('Date').dayOfYear()
```

17.8.7 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 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

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

Syntax

```
format(pattern)
```

Parameter	Description
`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'))
```

17.8.8 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.

See also *hourOfDay*.

Syntax

```
hour()
```

Examples

```
$('Date').hour()
```

17.8.9 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.

See also *hour*.

Syntax

```
hourOfDay()
```

Examples

```
$('Date').hourOfDay()
```

17.8.10 minute

Returns the minute within the hour.

Syntax

```
minute()
```

Examples

```
$('Date').minute()
```

17.8.11 millisecond

Returns the millisecond within the second.

Syntax

```
millisecond()
```

Examples

```
$('Date').millisecond()
```

17.8.12 month

Returns the month of a Date as an integer starting from 0 (January).

Syntax

```
month()
```

Examples

```
$('Date').month()
```

17.8.13 quarter

Returns the quarter of a Date as an integer starting from 0 (Q1).

Syntax

```
quarter()
```

Examples

```
$('Date').quarter()
```

17.8.14 second

Returns the second within the minute.

Syntax

```
second()
```

Examples

```
$('Date').second()
```

17.8.15 semester

Returns the semester of a Date as an integer starting from 0 (Q1).

Syntax

```
semester()
```

Examples

```
$('Date').semester()
```

17.8.16 time

Returns the number of milliseconds since January 1, 1970, 00:00:00 GMT (epoch time).

Syntax

```
time()
```

Examples

```
$('Date').time()
```

17.8.17 weekday

Returns a boolean value indicating whether the date denotes a weekday (between Monday and Friday inclusively).

Syntax

```
weekday()
```

Examples

```
$('Date').weekday()
```

17.8.18 weekend

Returns a boolean value indicating whether the date denotes a weekend (either Sunday or Saturday).

Syntax

```
weekend()
```

Examples

```
$('Date').weekend()
```

17.8.19 weekOfMonth

Returns the week of month from a date as an integer starting from 1.

Syntax

```
weekOfMonth()
```

Examples

```
$('Date').weekOfMonth()
```

17.8.20 weekOfYear

Returns the week of year from a date as an integer starting from 1.

Syntax

```
weekOfYear()
```

Examples

```
$('Date').weekOfYear()
```

17.8.21 year

Returns the year value.

Syntax

```
year()
```

Examples

```
$('Date').year()
```

17.9 Geo Value Methods

17.9.1 latitude

Returns the latitude of a point value as a decimal value.

See also *longitude*.

Syntax

```
latitude()
```

Examples

```
$('COORDINATE').latitude()
```

17.9.2 longitude

Returns the longitude of a point value as a decimal value.

See also *latitude*.

Syntax

```
longitude()
```

Examples

```
$('COORDINATE').longitude()
```

17.10 Measurement Unit Methods

17.10.1 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).

See also *toUnit*.

Syntax

```
unit([value])
```

Parameter	Description
`value`	The optional new measurement unit (must be a string).

Examples

```
$('HEIGHT').unit() // returns 'cm'
$('HEIGHT').unit('m') // returns the same value, but with a measurement unit of 'm'
```

17.10.2 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.

See also *unit*.

Syntax

```
toUnit(newUnit)
```

Parameter	Description
`newUnit`	The new measurement unit (must be a string).

Examples

Converts the value from meters to centimeters:

```
$('HEIGHT').unit('m').toUnit('cm')
```

Converts the value from its current unit to centimeters (the current unit is take from the unit property of the HEIGHT variable):

```
$('HEIGHT').toUnit('cm')
```

Global Methods

Methods	Description
$, $val, $value	Return the variable value within the current value set.
$this	Return the variable value within the current view value set.
$join	Return the joined variable value referenced by a variable value within the current 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.
$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

Methods	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`, `CONTINUOUS`, 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

Methods	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.
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

Methods	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 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

Methods	Description
and	Applies the ternary AND logic on values.
compare	Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.
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

Methods	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.

Text Value Methods

Methods	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.
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.
concat	Returns the text type result of first operand concat second operand.
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 Datetime Value Methods

Methods	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

Methods	Description
longitude	Get the longitude of a point value.
latitude	Get the latitude of a point value.

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.