Opal Documentation

OBiBa

Apr 01, 2024

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.

CHAPTER ONE

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.

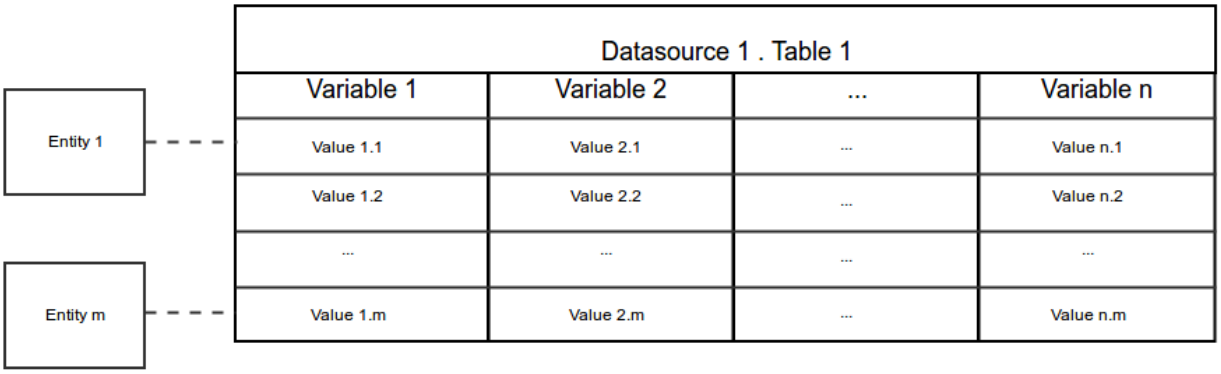
See detailed concepts and tutorials:

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:

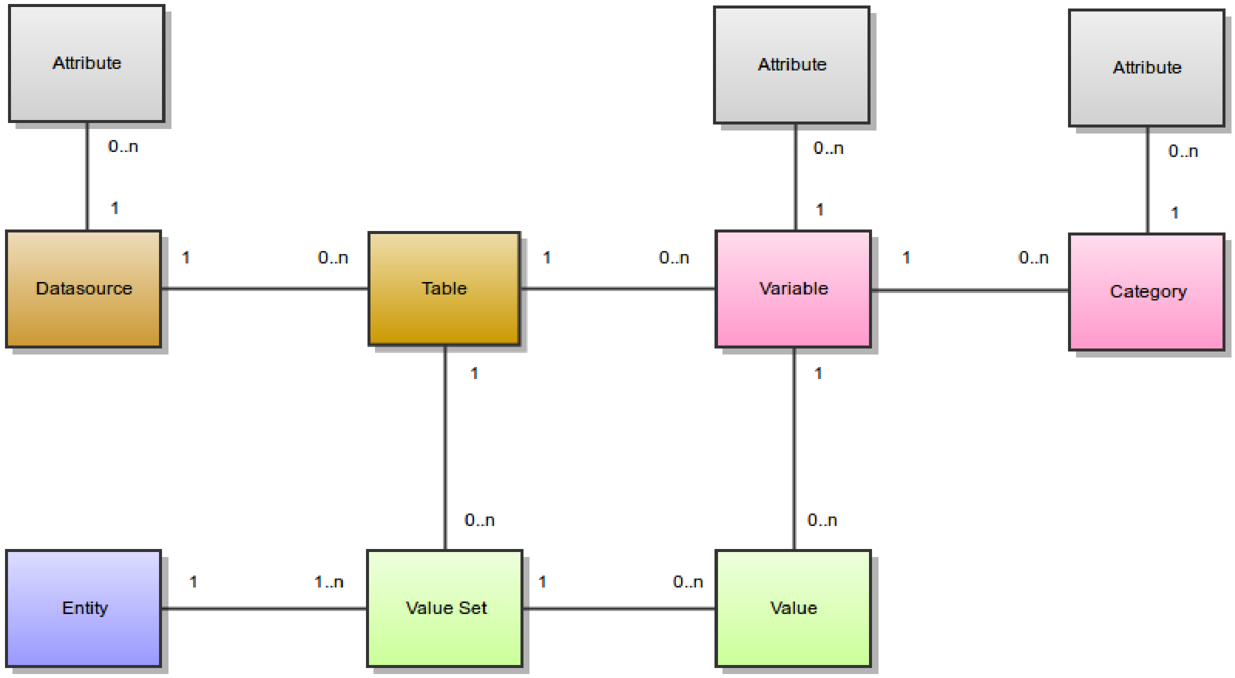
- the 'columns' are the variables,
- the 'rows' are the value sets for each entity,
- the 'cells' are the variable entity values.



Datasource 1 . Table 1

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

The following diagram shows the relationships between the different concepts:



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.

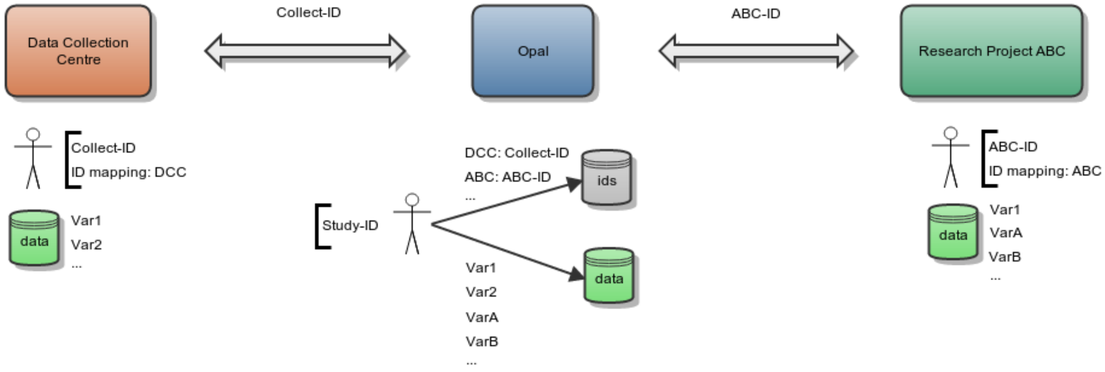
A value is always associated with a type and a data (or a sequence of data if the variable is `repeatable`).

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

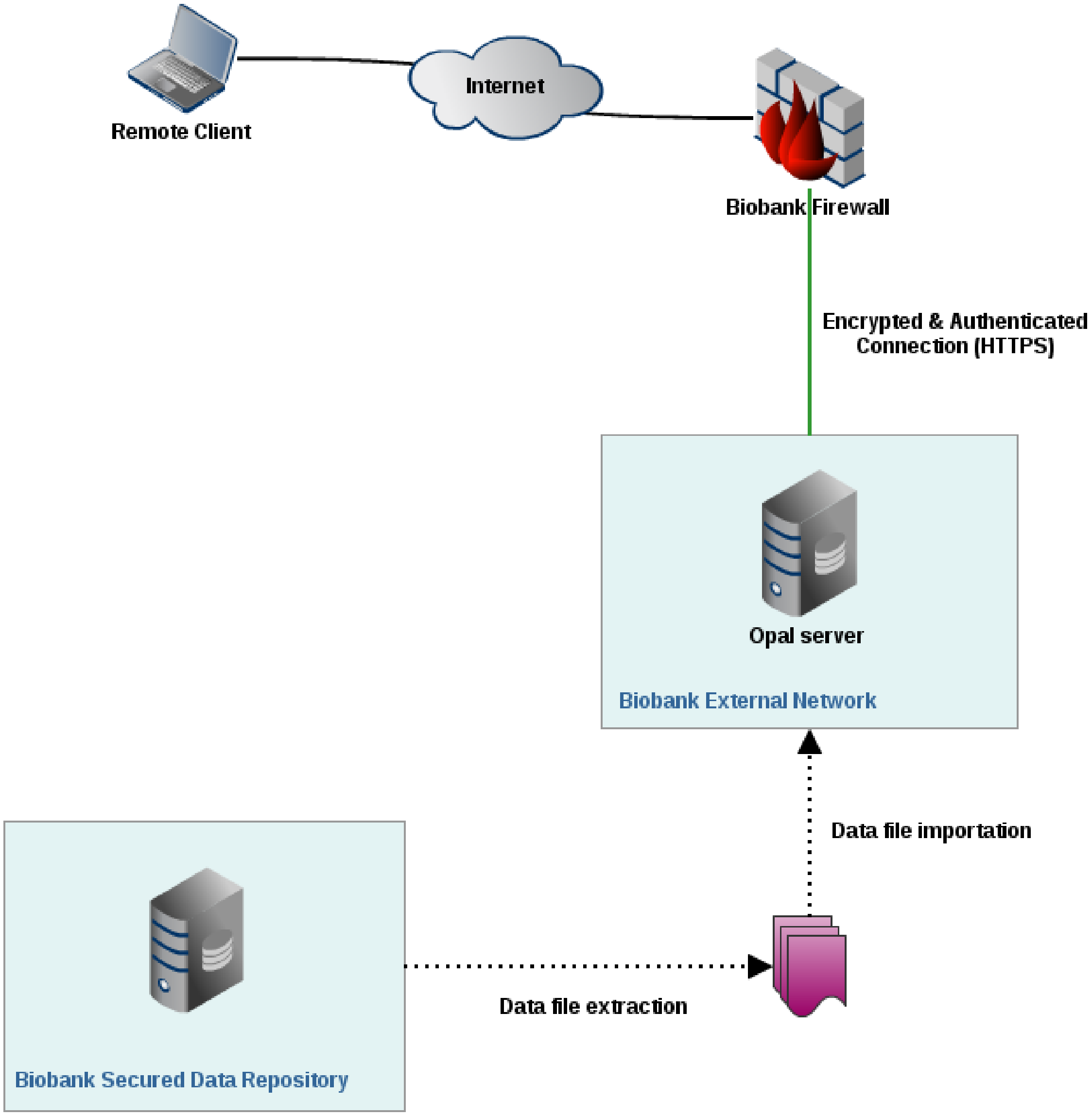
Opal in Study Networks

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:

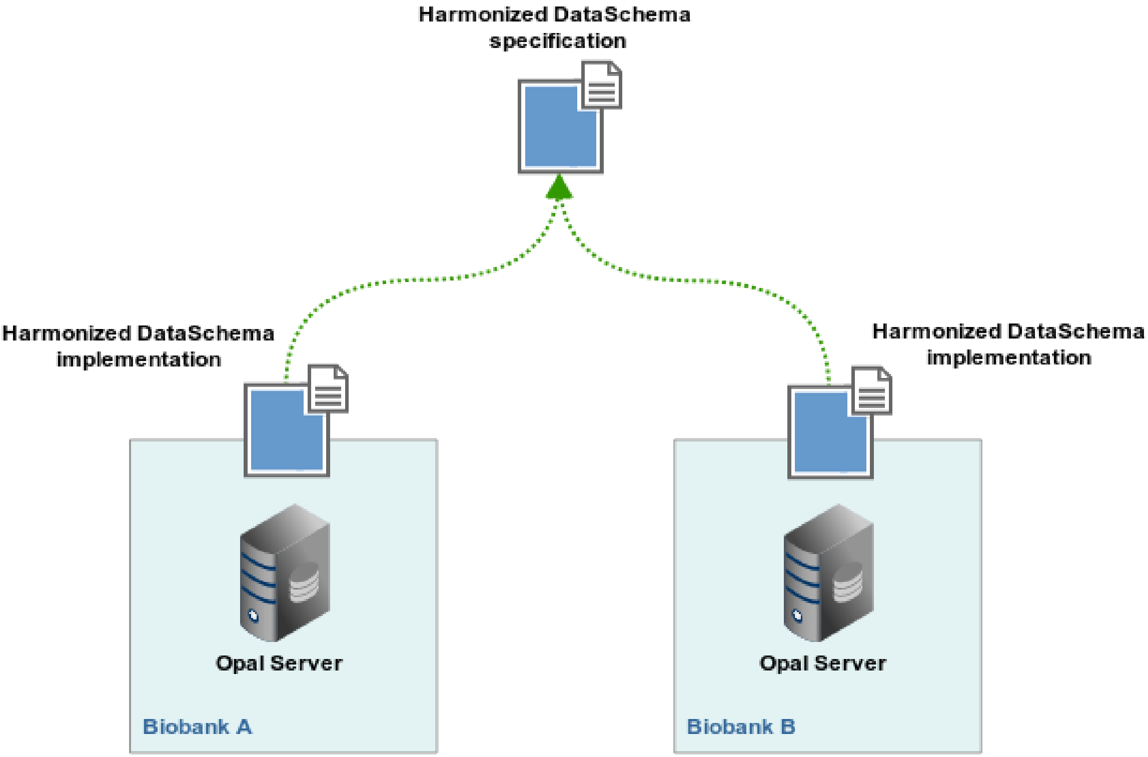
- The Opal server is hosted in a network that is protected by a firewall which only allows connections using encrypted (HTTPS) protocol through a specific port. Connections can also be restricted to specific remote clients.
- Opal application itself requires user authentication. Data hosted by Opal are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

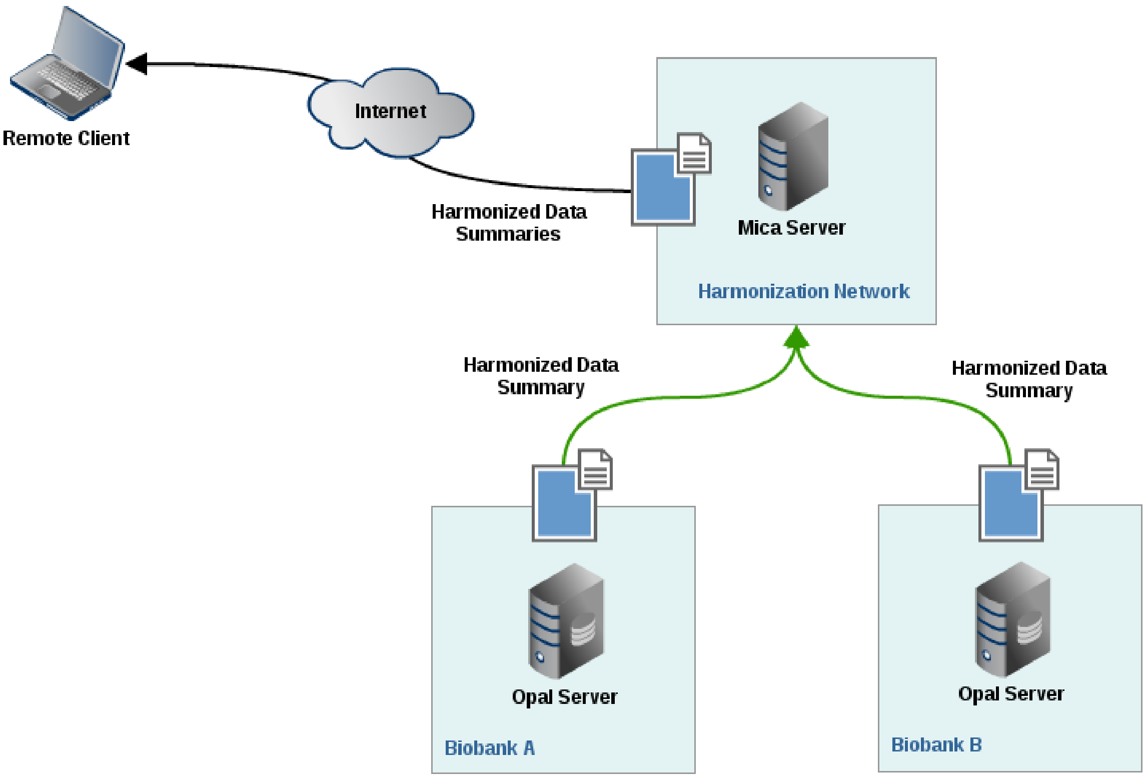

Harmonized DataSchema specification
Harmonized DataSchema implementation
Harmonized DataSchema implementation
Opal Server
Biobank A
Opal Server
Biobank B

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.

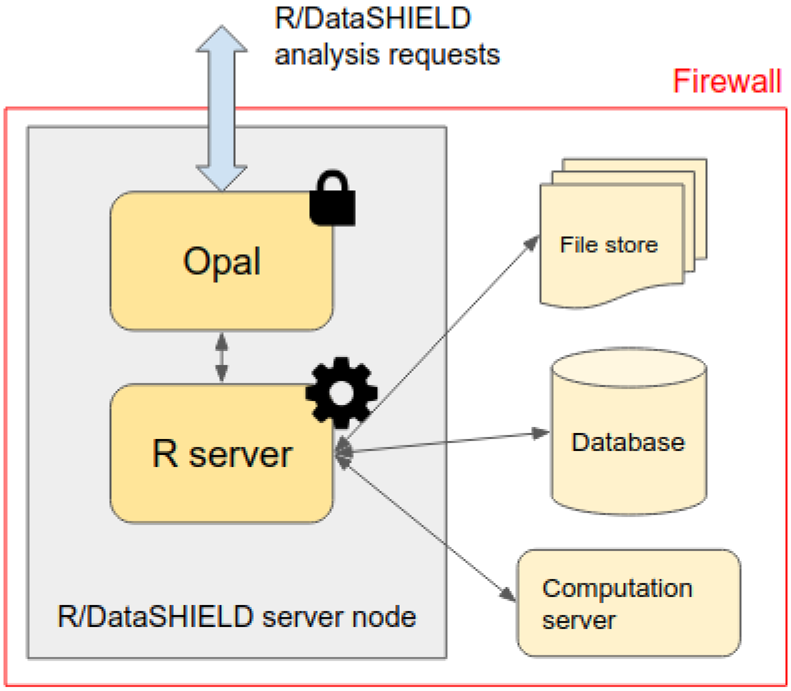
1.4 Resources

Developing and implementing new algorithms to perform advanced data analyses in R or under the DataSHIELD framework is a current active line of research. However, the analysis of big data within R/DataSHIELD has some limitations. Some of them are related to how data is managed in the Opal's database and others are related to how to perform statistical analyses of big data within the R environment. Opal databases are for general purpose and do not properly manage large amounts of information and, second, it requires moving data from original repositories into Opal which is inefficient (this is a time, CPU and memory consuming operation) and is difficult to maintain when data are updated. We have overcome the problem related to R/DataSHIELD big data management by developing a new data infrastructure within Opal: **the resources**.

1.4.1 Concept

Resources are datasets or computation units which location is described by a URL and access is protected by credentials. When assigned to a R/DataSHIELD server session, remote big/complex datasets or high performance computers are made accessible to data analysts.

Instead of storing the data in Opal's database, only the way to access them is to be defined: the datasets are kept in their original format and location (a SQL database, a SPSS file, R object, etc.) and are read directly from the R/DataSHIELD server-side session. Then as soon as there is a R reader for the dataset or a connector for the analysis services, a resource can be defined. Opal takes care of the permissions (a DataSHIELD user cannot see the resource's credentials) and of the resources assignment to a R/DataSHIELD session.



Note that Opal does not require the usage of a database backend (such as MySQL/MariaDB or MongoDB) to manage the resources: this simplifies a lot the deployment of Opal (memory footprint is also drastically reduced) as data access is delegated to the R server.

1.4.2 Types

The data format refers to the intrinsic structure of the data. A very common family of data formats is the tabular format which is made of rows (entities, records, observations etc.) and columns (variables, fields, vectors etc.). Examples of tabular formats are the delimiter-separated values formats (CSV, TSV etc.), the spreadsheet data formats (Microsoft Excel, LibreOffice Calc, Google Sheets etc.), some proprietary statistical software data formats (SPSS, SAS, Stata etc.), the database tables that can be stored in structured database management systems that are row-oriented (MySQL, MariaDB, PostgreSQL, Oracle, SQLite etc.) or column-oriented (Apache Cassandra, Apache Parquet, MariaDB ColumnStore, BigTable etc.), or in semi-structured database management systems such as the documented-oriented databases (MongoDB, Redis, CouchDB, Elasticsearch etc.).

When the data model is getting complex (data types and objects relationships), a domain-specific data format is sometimes designed to handle this complexity so that statistical analysis and data retrieval can be executed as efficiently as possible. Examples of domain-specific data formats are encountered in the omic or geospatial fields of research. A data format can also include some additional features such as data compression, encoding or encryption. Each data format requires an appropriate reader software library or application to extract the information or perform data aggregation or filtering operations.

We have prepared a test environment, with the Opal implementation of the resources and an appropriate R/DataSHIELD configuration that is available at: opal-demo.obiba.org, more specifically in the RSRC project (use administrator/password to login). As it can be seen this example project, the data storage can simply be a file to be accessed directly from the host's file system or to be downloaded from a remote location. More advanced data storage systems are software applications that expose an interface to query, extract or analyze the data. These applications can make use of a standard programming interface (e.g. SQL) or expose specific web services (e.g. based on the HTTP communication protocol) or provide a software library (in different programming languages) to access the data. These different ways of accessing the data are not exclusive from each other. In some cases the micro-data cannot be extracted, only computation services that return aggregated data are provided. The data storage system can also apply security rules, requiring authentication and proper authorizations to access or analyze the data.

1.4.3 Definition

We call *resource* this data or computation access description. A resource will have the following properties:

- the **location** of the data or of the computation services,
- the **data format** (if this information cannot be inferred from the location property),
- the **access credentials** (if some apply).

The resource location description will make use of the web standard described in the RFC 3986 "Uniform Resource Identifier (URI): Generic Syntax". More specifically, the Uniform Resource Locator (URL) specification is what we need for defining the location of the data or computation resource: the term Uniform allows to describe the resource the same way, independently of its type, location and usage context; the term Resource does not limit the scope of what might be a resource, e.g. a document, a service, a collection of resources, or even abstract concepts (operations, relationships, etc.); the term Locator both identifies the resource and provides a means of locating it by describing its access mechanism (e.g. the network location). The URL syntax is composed of several parts:

- a *scheme*, that describes how to access the resource, e.g. the communication protocols "https" (secured HTTP communication), "ssh" (secured shell, for issuing commands on a remote server), or "s3" (for accessing Amazon Web Service S3 file store services),
- an *authority* (optional), e.g. a server name address,
- a *path* that identifies/locates the resource in a hierarchical way and that can be altered by query parameters.

The resource's data format might be inferred from the path part of the URL, by using the file name suffix for instance. Nevertheless, sometimes it is not possible to identify the data format because the path could make sense only for the data storage system, for example when a file store designates a document using an obfuscated string identifier or when a text-based data format is compressed as a zip archive. The format property can provide this information.

Despite the authority part of the URL can contain some user information (such as the username and password), it is discouraged to use this capability for security considerations. The resource's credentials property will be used instead, and will be composed of an identifier sub-property and a secret sub-property, which can be used for authenticating with a username/password, or an access token, or any other credentials encoded string. The advantage of separating the credentials property from the resource location property is that a user with limited permissions could have access to the resource's location information while the credentials are kept secret.

Once a resource has been formally defined, it should be possible to build programmatically a connection object that will make use of the described data or computation services. This resource description is not bound to a specific programmatic language (the URL property is a web standard, other properties are simple strings) and does not enforce the use of a specific software application for building, storing and interpreting a resource object.

1.4.4 resourcer R Package

The resourcer package is an R implementation of the data and computation resources description and connection. It is reusing many existing R packages for reading various data formats and connecting to external data storage or computation servers. The resourcer package role is to interpret a resource description object to build the appropriate resource connection object. Because the bestiary of resources is very wide, the resourcer package provides a framework for dynamically extending the interpretation capabilities to new types of resources.

Opal uses the Resource Forms API to dynamically discover new type of resources installed in the R server and build for each of them a user-friendly interface to capture the domain specific resource details.

CHAPTER

TWO

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	8	OpenJDK 8 downloads	Java runtime environment
MySQL	>= 5.5.x	MySQL downloads	Database engine
MongoDB	<= 6.0.x	MongoDB Community downloads	Database engine
R	>= 4.x	R downloads	Statistical analysis engine

2.2 Install

Opal is distributed as a Debian/RPM package, as a zip file and as a Docker image. 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 *Server launch*.

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 *Server launch*.

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 *Server launch*.

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

2.2.4 Docker Image Installation

OBiBa is an early adopter of the Docker technology, providing its own images from the Docker Hub repository.

A typical docker-compose file (including a MongoDB database and a MySQL database, a DataSHIELD ready R server and all useful plugins) would be:

```
version: '3'
services:
  opal:
    image: obiba/opal:latest
    ports:
      - "8880:8080"
    links:
```

(continues on next page)

(continued from previous page)

```
      - rock
      - mongo
      - mysqldata
    environment:
      #- JAVA_OPTS=-Xms1G -Xmx8G -XX:+UseG1GC
      - OPAL_ADMINISTRATOR_PASSWORD=${OPAL_ADMINISTRATOR_PASSWORD}
      - MONGO_HOST=mongo
      - MONGO_PORT=27017
      - MYSQLDATA_HOST=mysqldata
      - MYSQLDATA_DATABASE=${MYSQLDATA_DATABASE}
      - MYSQLDATA_USER=${MYSQLDATA_USER}
      - MYSQLDATA_PASSWORD=${MYSQLDATA_PASSWORD}
      - ROCK_HOSTS=rock:8085
    volumes:
      - /some/path/opal:/srv
  mongo:
    image: mongo:6.0
  mysqldata:
    image: mysql
    environment:
      - MYSQL_DATABASE=${MYSQLDATA_DATABASE}
      - MYSQL_USER=${MYSQLDATA_USER}
      - MYSQL_PASSWORD=${MYSQLDATA_PASSWORD}
      - MYSQL_RANDOM_ROOT_PASSWORD=yes
  rock:
    image: obiba/rock:latest
```

The environment variables that are exposed by this image are:

Environment Variable	Description
`JAVA_OPTS`	Java VM arguments.
`OPAL_ADMINISTRATOR_PASSWORD`	Opal administrator password, required and set at first start.
`APP_URL`	Opal public URL (optional, see `org.obiba.opal.public.url` setting).
`APP_CONTEXT_PATH`	Opal server URL context (optional, see `org.obiba.opal.server.context-path` setting).
`MONGO_HOST`	MongoDB server host (optional).
`MONGO_PORT`	MongoDB server port, default is `27017`.
`MONGO_USER`	MongoDB server user (optional).
`MONGO_PASSWORD`	MongoDB server password (optional).
`MONGODATA_DATABASE`	MongoDB server data database name.
`MONGOIDS_DATABASE`	MongoDB server IDs database name (optional, ignored if a MySQL, MariaDB or PostgreSQL d
`MYSQLDATA_HOST`	MySQL server host for data storage (optional).
`MYSQLDATA_PORT`	MySQL server port.
`MYSQLDATA_DATABASE`	MySQL data database name.
`MYSQLDATA_USER`	MySQL data database user.
`MYSQLDATA_PASSWORD`	MySQL data database password.
`MYSQLIDS_HOST`	MySQL server host for IDs storage (optional).
`MYSQLIDS_PORT`	MySQL server port.
`MYSQLIDS_DATABASE`	MySQL IDs database name.
`MYSQLIDS_USER`	MySQL IDs database user.
`MYSQLIDS_PASSWORD`	MySQL IDs database password.
`MARIADBDATA_HOST`	MariaDB server host for data storage (optional).

continues

Table 1 – continued from previous page

Environment Variable	Description
MARIADBDATA_DATABASE	MariaDB data database name.
MARIADBDATA_USER	MariaDB data database user.
MARIADBDATA_PASSWORD	MariaDB data database password.
MARIADBIDS_HOST	MariaDB server host for IDs storage (optional, ignored if a MySQL one is defined).
MARIADBIDS_DATABASE	MariaDB IDs database name.
MARIADBIDS_USER	MariaDB IDs database user.
MARIADBIDS_PASSWORD	MariaDB IDs database password.
POSTGRESDATA_HOST	PostgreSQL server host for data storage (optional).
POSTGRESDATA_DATABASE	PostgreSQL data database name.
POSTGRESDATA_USER	PostgreSQL data database user.
POSTGRESDATA_PASSWORD	PostgreSQL data database password.
POSTGRESIDS_HOST	PostgreSQL server host for IDs storage (optional, ignored if a MySQL or MariaDB one is defin
POSTGRESIDS_DATABASE	PostgreSQL IDs database name.
POSTGRESIDS_USER	PostgreSQL IDs database user.
POSTGRESIDS_PASSWORD	PostgreSQL IDs database password.
AGATE_URL	Agate server URL (optional).
AGATE_HOST	[Deprecated, use AGATE_URL] Agate server host (optional).
AGATE_PORT	[Deprecated, use AGATE_URL] Agate server port, default is 8444.
ROCK_HOSTS	Comma separated Rock R server URLs, for R server discovery (optional, but recommended).
ROCK_ADMINISTRATOR_USER	Default Rock server administrator user name (optional).
ROCK_ADMINISTRATOR_PASSWORD	Default Rock server administrator user password (optional).
ROCK_MANAGER_USER	Default Rock server manager user name (optional).
ROCK_MANAGER_PASSWORD	Default Rock server manager user password (optional).
ROCK_USER_USER	Default Rock server user user name (optional).
ROCK_USER_PASSWORD	Default Rock server user user password (optional).
R_REPOS	R CRAN repositories (optional, see org.obiba.opal.r.repos setting).

See also the Rock R server Docker documentation.

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.

2.3.4 Docker Distribution Upgrade

Change the docker image version and restart the docker container. If the opal-home directory was mounted in user space, it will be reused.

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.

Docker

When using a docker-compose configuration file, the start up command is:

```
docker-compose -f docker-compose.yml up -d
```

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

THREE

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.
`org.obiba.opal.server.context-path`	The context path when server is accessed at a subdirectory (for instance in `http://example.org/opal` the context path is `/opal`). Default is empty.

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 Apps Configuration

External applications can be discovered or can self-register. The following settings the apps management defaults.

Property	Description
`apps.registration.token`	Apps self-registration default token. Default is empty (self-registration not allowed). When configured from *Apps* administration page, this value is overridden.
`apps.registration.include`	White list rule to accept an app self-registration: (java) regular expression applied to app's server address. If not defined (default) all apps are filtered-in.
`apps.registration.exclude`	Black list rule to accept an app self-registration: (java) regular expression applied to app's server address. If not defined (default) no app is filtered-out.
`apps.discovery.interval`	Apps discovery scheduling in milliseconds. Default is `10000`.
`apps.discovery.rock.hosts`	Comma separated list of Rock R server URLs to discover on start up. Default is `localhost:8085`. See also *R Server Configuration*.

3.1.5 R Server Configuration

Opal is able to perform R queries by talking with a running R server. See the *R Server* documentation. The properties for connecting to the default Rock R server(s) are the following:

Property	Description
rock.default.administrator.username	Rock administrator user name. Default is administrator.
rock.default.administrator.password	Rock administrator user password. Default is password.
rock.default.manager.username	Rock manager user name.
rock.default.manager.password	Rock manager user password.
rock.default.user.username	Rock regular user name.
rock.default.user.password	Rock regular user password.
org.obiba.opal.r.endpoint	Enable/disable the plain R web service. When disabled, even the system administrator cannot interact directly with a plain R session. Use of the DataSHIELD web service is recommended instead. Default is true (enabled).
org.obiba.opal.r.sessionTimeout	Time in minutes after which an inactive R session will be automatically terminated (default is 4 hours).
org.obiba.opal.r.sessionTimeout.R	Time in minutes after which an inactive R session with **R context** will be automatically terminated (default is to fallback to org.obiba.opal.r.sessionTimeout).
org.obiba.opal.r.sessionTimeout.DataSHIELD	Time in minutes after which an inactive R session with **DataSHIELD context** will be automatically terminated (default is to fallback to org.obiba.opal.r.sessionTimeout).
org.obiba.opal.r.sessionTimeout.Import	Time in minutes after which an inactive R session with **Import context** will be automatically terminated (default is to fallback to org.obiba.opal.r.sessionTimeout).
org.obiba.opal.r.sessionTimeout.Export	Time in minutes after which an inactive R session with **Export context** will be automatically terminated (default is to fallback to org.obiba.opal.r.sessionTimeout).
org.obiba.opal.r.sessionTimeout.Report	Time in minutes after which an inactive R session with **Report context** will be automatically terminated (default is to fallback to org.obiba.opal.r.sessionTimeout).
org.obiba.opal.r.sessionTimeout.SQL	Time in minutes after which an inactive R session with **SQL context** will be automatically terminated (default is to fallback to org.obiba.opal.r.sessionTimeout).
org.obiba.opal.r.sessionTimeout.Analyse	Time in minutes after which an inactive R session with **Analyse context** will be automatically terminated (default is to fallback to org.obiba.opal.r.sessionTimeout).
org.obiba.opal.r.sessionTimeout.View	Time in minutes after which an inactive R session with **View context** will be automatically terminated (default is to fallback to org.obiba.opal.r.sessionTimeout).
org.obiba.opal.r.repos	The list of CRAN repositories from which R packages can be downloaded, comma separated. Default value is https://cloud.r-project.org,https://cran.obiba.org.

3.1.6 DataSHIELD Configuration

Some minimal default DataSHIELD infrastructure settings can be defined.

Property	Description
datashield.r.parser	DataSHIELD R parser version: v1 or v2 See DataSHIELD4J library documentation. Default is the latest.

3.1.7 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.
org.obiba.opal.security.login.pat.expiresIn	Time in days after which a personal access token is automatically removed. Default is -1 (i.e. tokens never expire).
org.obiba.opal.security.login.pat.activityTimeout	Time in days since last access after which a personal access token is marked as being inactive. This state can be reverted by user. Default is 60 (2 months).

3.1.8 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.publicUrl	Public address to create a link from Opal's user profile page to the Agate's one where personal information, password and two-factor authentication can be managed. Default is empty.
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.

3.1.9 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`.

3.1.10 Cross Site Resource Forgery (CSRF)

CSRF attacks can be mitigated by a built-in interceptor. Default behavior allows connections (http or https) from `localhost` and `127.0.0.1`. Requests from pages served by Opal should be allowed as well (https only), unless network settings or proxies modify or do not report the referer URL.

Property	Description
`csrf.allowed`	Comma separated list of client `host:port` explicitly allowed to connect to Opal server. Use `*` as a wildcard. Default is empty.

3.1.11 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`.
`productionMode`	When set to `false` the CSRF check is disabled and plugin jars conflict checks are skipped. Default value is `true`.

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
# =======================

[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]*
admin = *
```

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 =
```

3.4 Logging

The runtime messages can be configured in the **OPAL_HOME/conf/logback.xml** file. See Logback documentation.

By default, Logback is configured to output files in the **OPAL_HOME/logs** folder. The log files are:

- `opal.log`, contains the Opal application main log messages,
- `rest.log`, contains the web services specific log messages,
- `datashield.log`, contains the DataSHIELD activity log messages,
- `sql.log`, contains the SQL API specific log messages.

These log files can be downloaded from the web interface (**Administration > Java Virtual Machine > Logs** or **Administration > DataSHIELD > Logs**) or using the opalr R package.

3.4.1 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.

3.5 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 (requires `deflate` module) 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
```

(continues on next page)

(continued from previous page)

```
      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>
```

Recommended security headers are (to be added to the `apache2.conf` file, requires `headers` module):

```
# Security Headers, see https://securityheaders.com/
Header set Strict-Transport-Security "max-age=63072000"
Header set X-Frame-Options DENY
Header set X-XSS-Protection 1;mode=block
Header set X-Content-Type-Options nosniff
Header set Content-Security-Policy "frame-ancestors 'none'"
Header set Referrer-Policy "same-origin"
Header set Permissions-Policy "fullscreen=(self)"
Header onsuccess edit Set-Cookie ^(.+)$ "$1;HttpOnly;Secure;SameSite=Strict"
```

3.6 Proxy Configuration

Outbound connections may go through a proxy, depending on the host institution's network setup. It is possible to declare the proxy settings by modifying the `JAVA_OPTS` environment variable. As an example:

```
# without authentication
JAVA_OPTS=-Dhttp.proxyHost=127.0.0.1 -Dhttp.proxyPort=3020 -Xms1G -Xmx8G

# or with authentication
JAVA_OPTS=-Dhttp.proxyHost=127.0.0.1 -Dhttp.proxyPort=3020 -Dhttp.proxyUser=opal -Dhttp.
↪proxyPassword=xxxxxx -Xms1G -Xmx8G
```

CHAPTER

FOUR

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 mete-data 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

FIVE

R SERVER

Opal delegates many of its computation and data source connection capabilities to the attached R server(s).

See the Rock R server documentation to install and configure one or more instances of R server that Opal will use to balance the computation load.

Read also the *Apps* documentation to know how to interconnect Opal with some R servers.

5.1 R Server Scalability

Opal can interact with several R servers, grouped in a cluster. All R servers from a cluster are expected to be the same (version of R, R packages installed), i.e. Opal can create new R sessions in one or another. This allows **load balancing** between R servers, especially useful in a multi-user environment when computation intensive R operations are done (omics analysis for instance).

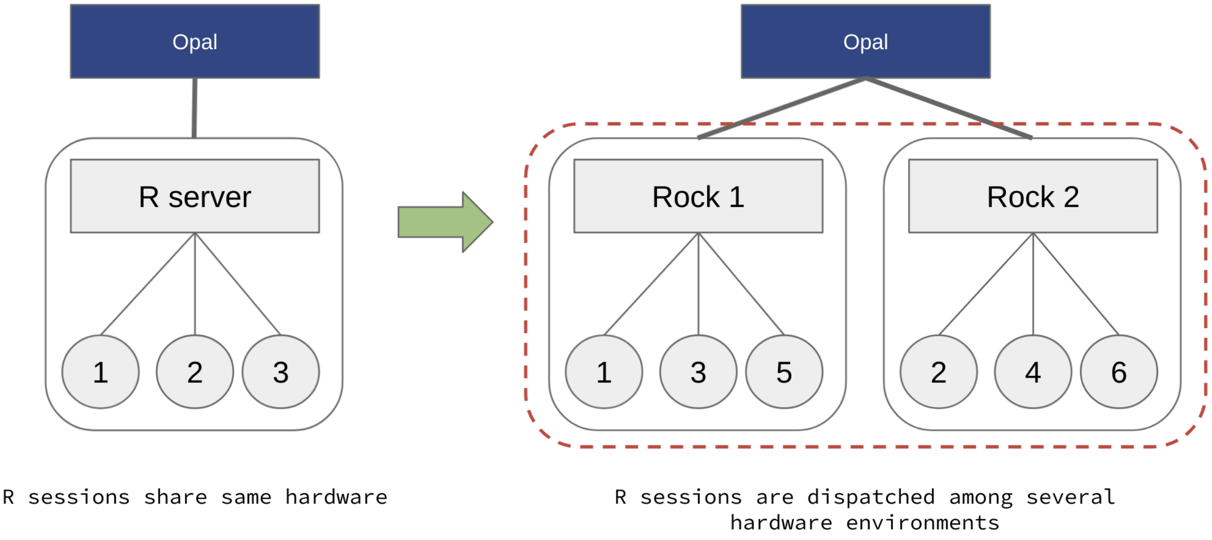

Opal
R server
1
2
3
R sessions share same hardware
Opal
Rock 1
1
3
5
Rock 2
2
4
6
R sessions are dispatched among several hardware environments

Fig. 1: Add more Rock servers and balance the load to scale R services horizontally.

5.2 R Server Profiles

Opal supports multiple clusters of R servers, each one is designated by a **profile** name. This allows to have different flavors of R servers living aside of each others: different versions of R, different R packages installed, in different versions. Such configuration is easy to setup when deploying Rock with Docker. As the footprint of a Rock server is small, using a containerized deployment it is possible to have several versions of R servers running on the same host.

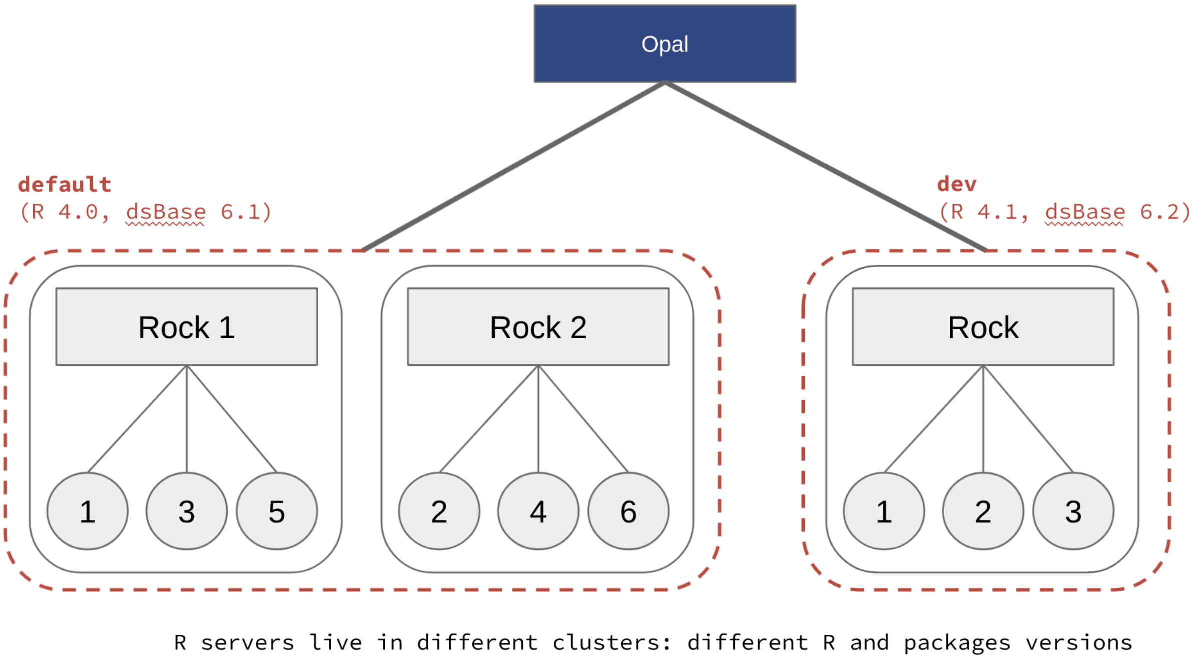
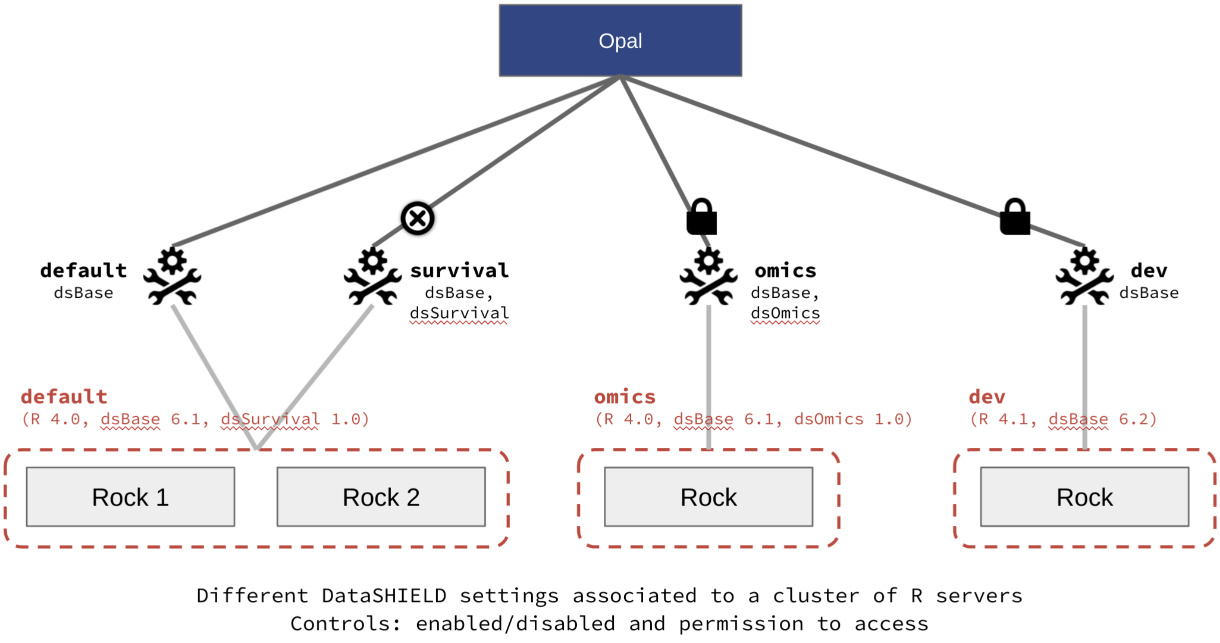


Fig. 2: Add more Rock server flavors and allow R users to select the appropriate one.

5.3 DataSHIELD Profiles

Based on Opal's capability to handle different profiles (clusters) of R servers, it is also possible to define several DataSHIELD profiles. A DataSHIELD profile is a R server profile combined with a DataSHIELD configuration (allowed functions, options and permissions). DataSHIELD users can then decide in which environment their analysis are to be performed, for a better reproducible science. DataSHIELD administrators can also fine tune the DataSHIELD configurations for advanced usages. Such configuration is easy to setup when deploying Rock with Docker, eliminating the burden of dealing with several versions of R and packages, and of installing potentially hundreds of R packages.



Different DataSHIELD settings associated to a cluster of R servers
Controls: enabled/disabled and permission to access

Fig. 3: Define different DataSHIELD configurations using profiles.

CHAPTER

SIX

COOKBOOK INTRODUCTION

This Cookbook intends to promote good practices when using Opal for the most common use cases.

6.1 Importing Data

Importing data, to make them available for other services (R/DataSHIELD analysis, reporting, making a data web portal etc.), is a fundamental feature of a data warehouse. Opal supports many different import formats from various sources (files, remote servers). There are also many different ways of triggering an import task (manually or using a script). Opal also offers to have a rich description of the data: data dictionary can contain useful information about the data origin and constitutes the first step to build a web data portal useful for researchers.

6.1.1 How to Import Data from Files

Importing SPSS, SAS, Stata Data

Unlike the CSV file format, the SPSS/SAS/Stata file formats contain their own data dictionary. Opal uses the haven R package (and then require a functional R server) to import data in these formats: value types, categories, missing values etc. are extracted from the R data structure read by *haven*.

Procedure

Note:

1. Go to the destination project's tables page,
2. Select **Import**,
3. Select the Data Format **SPSS (R)**, or **SAS (R)** etc., follow instructions (upload data file, set options etc.) and launch the import task,
4. When the import task is completed, go to the new table's page.

The table is created/updated with the imported data

Importing CSV Data

The following applies to any delimited textual file formats, CSV (comma delimiter) being the most common one.

Opal supports two CSV importers:

- **CSV**, which is the built-in one,
- **CSV (R)**, available as a plugin (*opal-datasource-readr*, see *Plugins* administration), based on the readr R package.

The main difference between these two CSV importers is how the data types are handled. **CSV (R)** is recommended for its capability to detect data types.

CSV (R) Importer

This R-based importer will attempt to detect the data types by reading the first 1000 rows. In case a variable has no values in these first rows, the **text** value type will be used.

Procedure

Note:

0. Preliminary: having the *opal-datasource-readr* plugin installed and a functional R server,
1. Go to the destination project's tables page,
2. Select **Import**,
3. Select the Data Format **CSV (R)**, follow instructions (upload CSV file, set options etc.) and launch the import task,
4. When the CSV (R) import task is completed, go to the new table's page.

The table is created/updated with the imported data

CSV Importer

In the case of **CSV** importer, the data types are NOT guessed from the provided CSV file:

- Either the destination table already exists and then the variable value types are the ones declared (if there is an inconsistency with the provided data, import will fail),
- Or all the value types are **text**.

Procedure

Setting up the destination table variables prior to the **CSV** import can be counter intuitive and error prone (missing variables, wrong data types): this is NOT a recommended procedure. The recommended approach is to import first the CSV data as-is and then make a view (i.e. a logical table) to transform variable value types to the correct ones.

Note:

1. Go to the destination project's tables page,
2. Select **Import**,
3. Select the Data Format **CSV**, follow instructions (upload CSV file, set options etc.) and launch the import task,
4. When the CSV import task is completed, go to the new table's page (note that all variables have **text** type),
5. Make a view based on the imported table: *How to Transform Tables with Views*.

The table is created/updated with the imported data, and is accessed through a view

6.1.2 How to Import Data from R

If the Opal server's data importers are not sufficient (unsupported data format, missing data extraction options etc.), the recommended alternative is to use a R script (which execution can be automated to update data) as follows:

Note:

0. Preliminary: install opalr R package
1. Connect to Opal server using `opal.login()`
2. Load and prepare data in R as a `tibble` object
3. [optional] Fine tune data dictionary using `attributes()` or `dictionary.apply()` or `dictionary.annotate()`
4. Save data using `opal.table_save()`
5. [optional] Update data dictionary using `opal.table_dictionary_get()`

The table is created/updated with the imported data and is to be accessed directly or through a view

Step 1 - Prerequisites

Install R Packages

Opal is a server application. The client R script will connect the Opal server. Then the prerequisites are:

- Server: having Opal connected to a functional R server,
- Client: having the opalr R package installed, and data sources accessible.

See also the *Using R* documentation.

Connect with Opal Server

Your script must start with:

```
# load opal library
library(opalr)
# connect to the opal server
o <- opal.login(...)
```

See the opal.login() documentation for more details about credentials.

Prepare Project

If the destination project does not exist yet, it is possible to create it using R:

```
# create a new project with a database backend for storing tables' data
opal.project_create(o, "myproject", database = TRUE)
```

You can find more information in the Opal Projects vignette.

Step 2 - Prepare Data in R

Load Data in R

There are many ways of having data available in R whether the source is a file, a database, a remote service etc. Some reference manuals can be found at:

- R Data Import/Export
- R for Data Science: Data import
- rio: A Swiss-Army Knife for Data I/O
- …

Make sure your data are tidy, and identify which column holds the identifiers. The expected R data structure for data import into Opal is a tibble.

[optional] Prepare Data Dictionary

The data dictionary can be fully or partially described directly in the tibble that will be imported.

Raw R Attributes

The data dictionary can be fine-tuned before saving the data into a Opal table. As an example, a R column of type `double` can be saved as a variable with the `integer` value type in place of the default `decimal` one. This is done by setting the R vector attributes with some Opal keys.

The R attribute keys that can be used are:

R attribute	Description
`opal.value_type`	Variable value type, as described in the *Variables and Categories* section.
`opal.unit`	The measurement unit of the variable values.
`opal.referenced_entity_type`	The type of the entity referred when variable values are identifiers.
`opal.mime_type`	The mime type of the variable values.
`opal.repeatable`	Whether the variable has repeated values. True when value is “1”, false otherwise.
`opal.occurrence_group`	Name of the occurrence group, when several variables are repeated together.
`opal.index`	Position in the variables list, for ordering.

For instance, the column *cyl* will be interpreted as a vector of integer values at *data* importation time:

```r
data <- tibble::as_tibble(mtcars)
# apply 'opal.value_type' attribute to 'cyl' column
attributes(data$cyl) <- list(opal.value_type = 'integer')
```

Another example makes a numerical variable with categories in Opal from a factor column in R:

```r
data <- tibble::as_tibble(mtcars)
# make column a factor, each level will be a category
data$cyl <- as.factor(data$cyl)
# append 'opal.value_type' attribute to 'cyl' column
attributes(data$cyl) <- append(attributes(data$cyl), list(opal.value_type = 'integer'))
```

Full Data Dictionary

Another approach is to apply the full data dictionary (same structure as in the `Excel template`) to the tibble to be saved. Use the dictionary.apply() for that purpose.

It is not necessary to use Excel to define this data dictionary:

```r
data <- tibble::as_tibble(mtcars)
variables <- tibble::tribble(
  ~name, ~valueType, ~`label:en`,  ~`Namespace::Name`, ~unit, ~repeatable, ~index,
  "mpg", "decimal", "Mpg label",  "Value1", "years", 0, 1,
  "cyl", "integer", "Cyl label",  "Value2", "kg/m2", 0, 2,
  "disp", "decimal", "Disp label", NA, NA, 1, 3
)
categories <- tibble::tribble(
  ~variable, ~name, ~missing, ~`label:en`,
  "cyl", "4", 0, "Four",
  "cyl", "6", 0, "Six",
  "cyl", "8", 1, "Height"
)
data <- dictionary.apply(data, variables, categories)
```

Taxonomy Term Annotations

To annotate one or more variables with a taxonomy term without having to define a full data dictionary, see the dictionary.annotate() documentation.

```
# annotate some variables with a taxonomy term
data <- dictionary.annotate(data,
  variables = c("A_SDC_EDU_LEVEL", "A_SDC_EDU_LEVEL_AGE"),
  taxonomy = "Mlstr_area",
  vocabulary = "Sociodemographic_economic_characteristics",
  term = "Education")
```

Step 3 - Import Data

Saving the data into a Opal table is as simple as:

```
# save 'data' tibble into 'mytable' table, using 'id' column to provide identifiers
opal.table_save(o, data, project = "myproject", table = "mytable", id.name = "id")
```

See the opal.table_save() documentation for more details about saving operation options.

You can follow the import task progress in the project's **Tasks** page.

Step 4 - [optional] Update Data Dictionary

The data dictionary in Opal is the rich description of the data.

Table Dictionary

After data have been saved the data dictionary can be amended, except the variable value types. See previous section (*Prepare Data Dictionary*) to control value types at importation time.

Other data dictionary properties and attributes can be set using the same data structure as in the `Excel template`, expressed in R.

See the opal.table_dictionary_update() documentation (that can be usefully combined with opal.table_dictionary_get()).

As an example the following data dictionary defined in R is applied to an Opal table:

```
variables <- tibble::tribble(
  ~name, ~valueType, ~`label:en`,  ~`Namespace::Name`, ~unit, ~repeatable, ~index,
  "mpg", "decimal", "Mpg label",  "Value1", "years", 0, 1,
  "cyl", "integer", "Cyl label",  "Value2", "kg/m2", 0, 2,
  "disp", "decimal", "Disp label", NA, NA, 1, 3
)
categories <- tibble::tribble(
  ~variable, ~name, ~missing, ~`label:en`,
  "cyl", "4", 0, "Four",
  "cyl", "6", 0, "Six",
  "cyl", "8", 1, "Height"
)
opal.table_dictionary_update(o, "myproject", "mytable", variables, categories)
```

View Dictionary

When data type has not been specified before the import and needs to be changed, an Opal view can transform values on the fly. See the *How to Transform Tables with Views* for making a view based on the imported table using R.

6.1.3 How to Transform Tables with Views

A table is the result of importing data. Depending on the source of the data, the value types of the variables may not be correct (e.g. **text** instead of **integer**) and the data dictionary may be missing or incomplete (e.g. CSV provides only variable names).

Views are logical tables that can be used both for transforming data (e.g. recoding) and enhancing the data dictionary. The advantage of using a view instead of modifying the imported table's variables is that the table can be dropped, data can be updated/imported again without affecting the view's data dictionary.

From a Table

This is a manual operation that can be very straightforward.

Procedure

Note:

1. Go to the table's page
2. Select variables (either all or only the ones of interest)
3. Select **Add to view**

The view is created (or updated if it already exists) with the selected variables.

From a Variable

This operation allows to fine tune the transformation script, useful when setting categories from an originally continuous variable.

Procedure

Note:

1. Go to the variable's page
2. Select from the **Derive** menu the appropriate derivation operation
3. Follow derivation wizard instructions

The view is created (or updated if it already exists) with the selected variable.

Using Excel

It can be convenient for batch processing and for Excel lovers, to prepare the view using an Excel file. This file will both contains the variable descriptions and the data transformation scripts.

- Either follow the instructions in the `Excel template` to make your data dictionary from scratch,
- Or download the table's dictionary to start from a prefilled Excel file.

The derivation script is to be defined in the **script** column (see *Magma JS Introduction*).

Procedure

Note:

1. Go to the project's tables page
2. Select **Add Table > Add view…**
3. Upload the Excel file and select the table on which the view is based

The view is created with the provided variables.

Using R

The opalr R package is very powerful for interacting with a Opal server.

A view can be created with the opal.table_create() function and its data dictionary updated with the opal.table_dictionary_update() function.

See also data dictionary management examples from the Opal Projects vignette.

Procedure

Note:

0. Preliminary: install opalr R package
1. Connect to Opal server using `opal.login()`
2. Make the view's data dictionary in R
3. Create the view with `opal.table_create()` or update the view with `opal.table_dictionary_update()`

The view is created or updated with the provided variables.

6.1.4 How to NOT Import Data

There are different reasons why one would not import data into Opal:

- data are too large (timepoint series, genomes etc.)
- data are not tabular (linkage, multidimensional etc.)
- data cannot be extracted because they are linked to their computation environment (HPC, big data cluster etc.)
- data are already stored in a database that is kept updated
- data are not data (!) but computation services
- …

Opal offers the alternative of using *Resources*, combined with the resourcer R package. See also the chapter Orchestrating privacy-protected big data analyses of data from different resources with R and DataSHIELD: The Resources.

The general procedure is the following:

Note:

1. [optional] Design your own resource resolver/client in a R package, if the **resourcer** does not cover your needs, and make sure the R server(s) have this package installed
2. Declare the resources in a Opal project and apply appropriate access permissions
3. Use the resource from R/DataSHIELD in a R server

Step 1 - [optional] Design your Resource Resolver Package

The resourcer R package provides some of the most common resource resolvers (tidy files, databases and shell). Depending on the nature of your data and/or the location of these data, it may be necessary to design your own R package that extends the **resourcer** capabilities.

The dsOmics R/DataSHIELD package is an example of an R/DataSHIELD package that provides its own resource Resolvers. More specifically, *dsOmics* uses the Genomic Data Structure (GDS) file format for storing large, higly dimensional data on which computation can be done with a low memory footprint. It can also convert a Variant Call Format (VCF) file to a GDS one. As a reference, see the example code:

- GDSFileResourceResolver.R implements the GDS/VCF file resolver class,
- GDSFileResourceClient.R implements the GDS/VCF file client that downloads the file and establishes a GDS connection object,
- zzz.R registers the package's specific resolvers on library load,
- resource.js declares and documents the handled resources, so that Opal can discover them, generate corresponding resource creation forms, and load the R library before resource assignment in the R server.

Once the package is built, it needs to be installed in the Opal's R server(s).

Using Docker

In order to have a reproducible production environment, it is recommended to prepare a Docker image of the R server including your new package. See the *Step 1 - Prepare Docker Images* documentation in the R server profiles documentation.

From Administration Page

The package must be available in the official CRAN repository or in a GitHub source code repository or in Bioconductor repository.

Note:

1. Go to **Administration > R** page
2. Select the `default` R servers cluster
3. Select **Packages** tab
4. Press **Install** and follow instructions of *Install package*
5. Repeat for other R server clusters

Using R

In addition to the operations that can be performed from the web page, it is possible to upload a local R package to Opal directly using the oadmin.install_local_package() function. See also the other `oadmin.install*` functions.

Step 2 - Declare Resource in Opal

If the resource access is protected by credentials, it is recommended that these credentials have data read-only permission and/or have limited allowed operations. For example, use *Two-factor Authentication* to access a file stored in Opal, use a SQL view for a accessing a SQL database table etc.

From Project Page

Note:

1. Go to the project's page and select the **Resources** tab
2. Press **Add Resource** and select the *Category* (nature of the resource) and the *Type* (service or data format and location) of the resource
3. Fill in the connection form: *Parameters* and *Credentials*, and *Save*
4. [optional] From the created resource page, press **Test** to try resource assignment in the **default** R server. This will check if the R resource resolver can be found, but will not establish a connection with the resource

Using R

A resource can be added to a project by a simple function call, assuming that you know how to express the URL to the resource:

```
# load opal library
library(opalr)
# connect to the opal server
o <- opal.login(username = "administrator", password = 'password', url = 'https://opal-
↪demo.obiba.org')

# create an Opal file based resource
opal.resource_create(o, "RSRC", "CNSIM3",
  url = "opal+https://opal-demo.obiba.org/ws/files/projects/RSRC/CNSIM3.zip",
  format = "csv", secret = "EeTtQGIob6haio5bx6FUfVvIGkeZJfGq")

# to test the resource assignment and its resolution
opal.assign.resource(o, "client", "RSRC.CNSIM3")
opal.execute(o, "class(client)")

opal.logout(o)
```

Step 3 - Use the Resources

Unlike working with an Opal table (which R assignment is straightforward), when using an Opal resource reference the data/services are made accessible after the following operations:

1. assign Opal's resource reference to the R server and make a resource Client object: this object does not establish the connection with the resource yet but has the appropriate code to do it
2. Either coerce the resource Client object to a `data.frame` (if the data have a tabular representation) and/or execute Client's specific data extraction/computation functions (e.g. execute a remote shell command or perform some computation on a specific data structure etc.).

For coercing to the tabular representation of a resource, use the as.resource.data.frame() function (that is DataSHIELD compatible) on the resource Client object.

Using R

See *Using R* documentation for setting up the R client.

```
# load opal library
library(opalr)
# connect to the opal server
o <- opal.login(username = "administrator", password = 'password', url = 'https://opal-
↪demo.obiba.org')

# list resources in a project
opal.resources(o, "RSRC")

# assign a SQL database resource client
opal.assign.resource(o, "client", "RSRC.CNSIM1")
# coerce to a data.frame (tibble) and compute summary
```

(continues on next page)

(continued from previous page)

```
opal.assign.script(o, "data", quote(as.resource.data.frame(client)))
opal.execute(o, "summary(data)")

# assign a SSH resource client
opal.assign.resource(o, "sshClient", "RSRC.brge_plink")
# execute a shell command
opal.execute(o, "sshClient$exec('ls')")

opal.logout(o)
```

Using DataSHIELD

Given the power of the resources, DataSHIELD is a better analysis environment for securing the access to the resource's data and capabilities. See *Using DataSHIELD* documentation.

```
library(DSOpal)
library(dsBaseClient)
builder <- DSI::newDSLoginBuilder()
# connect to 'study1' on its 'default' profile
builder$append(server = "study1",  url = "https://opal-demo.obiba.org",
           user = "dsuser", password = "password")
logindata <- builder$build()
conns <- DSI::datashield.login(logins = logindata)

# list resources available
datashield.resources(conns)

# assign a resource client
datashield.assign.resource(conns, "client", "RSRC.CNSIM1")
# coerce to a (raw) data.frame and get summary
datashield.assign.expr(conns, "data", quote(as.resource.data.frame(client, strict =␣
↪TRUE)))
ds.summary("data")

datashield.logout(conns)
```

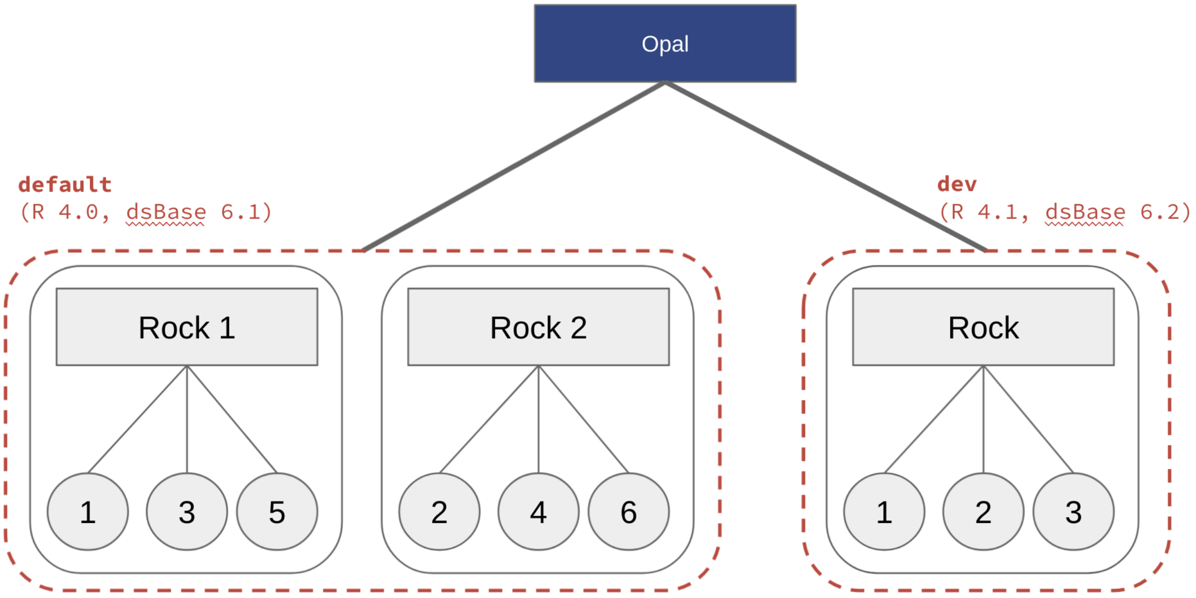
6.2 R/DataSHIELD

Data analyses happen in the R server backend, managed by Opal. Flexibility and scalibility are achieved by setting up several R/DataSHIELD profiles. Authentication and data access are controlled by Opal.

6.2.1 How to Set up R Server Profiles

Opal is able to connect to multiple R servers: see *R Server* documentation. The benefits of having multiple R servers are:

- Having different versions of R and/or R packages (i.e. profiles) installed, for reproducible science (see also *How to Set up R/DataSHIELD Client Profiles*),
- Balance the computation load other several R servers with same profile.



R servers live in different clusters: different R and packages versions

Opal has two different strategies for establishing connection with R servers (see *Apps* documentation):

- Self-registration, which is flexible and then appropriate for load balancing,
- Service discovery, which is preferred for managing multiple R server profiles.

Whatever the chosen registration strategy, the name of the R profile in Opal is the name of the `cluster` declared in the Rock R server configuration (see Cluster Node Configuration documentation).

Using the Docker technology, several R servers can run on the same host. An R server packaged in a Docker container is also easier to maintain, when R packages are to be updated or when a computation environment is to be restored. Therefore, the following instructions will recommend the Docker usage and more specifically the Docker Compose tool.

Step 1 - Prepare Docker Images

The following Rock R server Docker images are proposed:

Image	Description
obiba/rock	Rock R server application with R and useful R packages and system libraries. Everything you need for a standard R server connected to Opal (reporting, resources, analysis). Available tags are: - `latest`, `<rock_version>` (for instance `1.0`) - `<rock_version>-R<r-version>` (for instance `1.0-R4.1`).
datashield/rock-base	Based on `obiba/rock` image and includes the datashield/dsBase R package for basic DataSHIELD analysis. This is the recommended base image for the DataSHIELD users. Available tags are: - `latest`, `<dsBase_version>` (for instance `6.1`) - `<dsbase_version>-R<r_version>` (for instance `6.1-R4.1`). As a reference, see datashield/rock-base Dockerfile.
datashield/rock-mediation	Based on `datashield/rock-base` image and includes the datashield/dsMediation R package for mediation related DataSHIELD analysis. Available tags are: - `latest`, `<dsMediation_version>` (for instance `0.0.1`). As a reference, see datashield/rock-mediation Dockerfile.
datashield/rock-omics	Based on `datashield/rock-base` image and includes the isglobal-brge/dsOmics R package for omics related DataSHIELD analysis. Available tags are: - `latest`, `<dsOmics_version>` (for instance `1.0` or `1.0.7`). As a reference, see datashield/rock-omics Dockerfile.
datashield/rock-exposome	Based on `datashield/rock-base` image and includes the isglobal-brge/dsExposome R package for exposome related DataSHIELD analysis. Available tags are: - `latest`, `<dsExposome_version>` (for instance `1.0` or `1.0.1`). As a reference, see datashield/rock-exposome Dockerfile.

From these base images, it is possible to make your own, with additional R packages and system libraries installed. See for instance these demo images:

- obiba/rock-demo:geo Dockerfile installs geo system libraries and a DataSHIELD R package for geolocalized data analysis.

- obiba/rock-demo:survival Dockerfile installs the dsSurvival DataSHIELD package.

Step 2 - Docker Compose Configuration

Your Docker Compose configuration can include the Opal server but it is not mandatory, as the Opal server can be installed from a native package, whereas the multiple R servers will be started from docker images.

Different `datashield/rock-base` images can be used to expose different R packages versions. The DataSHIELD researcher can specify the appropriate profile name at connection time to ensure that the analysis envrionment is reproducible.

In the following example, several R servers will be accessible through their own port number, and the Opal server must be configured accordingly.

```
version: '3'
services:
    datashield:
        image: datashield/rock-base:latest
        ports:
            - ${PORT_DEFAULT}:8085
        environment:
            - ROCK_ID=${ROCK_ID}
            - ROCK_CLUSTER=default
    datashield-61:
        image: datashield/rock-base:6.1-R4.1
        ports:
            - ${PORT_BASE61}:8085
        environment:
            - ROCK_ID=${ROCK_ID}-base-6.1
            - ROCK_CLUSTER=base-6.1
```

Warning: Make sure there is always an R servers cluster name called `default`, which is the default profile name expected by Opal when none is specified.

Step 3 - Discover Rock R Servers in Opal

To configure Rock apps discovery in Opal, you can:

- Either set the `apps.discovery.rock.hosts` property in the **opal-config.properties** file, see *Apps Configuration* documentation. Opal server restart is then required.
- Or declare dynamically the new apps in the **Administration > Apps** page, *Discovery* section, see *Discovery* documentation. No Opal server restart is necessary.

You can then verify in the **Administration > R** page that several R servers are connected to, organized by cluster/profile. Each Rock R server reports host information (CPU cores, free memory) and the number of active R sessions.

Step 4 - Use R Profiles from R Client

In an R console, install the opalr R package. See also the *Using R* documentation.

The following script will list the R profiles on the opal-demo server:

```
# load opal library
library(opalr)
# connect to the opal server
o <- opal.login(username = "administrator", password = 'password', url = 'https://opal-
↪demo.obiba.org')

# list R profiles
opal.profiles(o)

# disconnect
opal.logout(o)
```

When an R operation requires to assign data and perform calculations in an R server, the R server profile can be specified at login time: a remote R session will be created in an R server from the named cluster.

```
# load opal library
library(opalr)
# connect to the opal server with profile 'geo'
o <- opal.login(username = "administrator", password = 'password', url = 'https://opal-
↪demo.obiba.org', profile = "geo")

# verify that 'dsGeo' is installed in this profile
oadmin.package_description(o, "dsGeo")

# assign some GPS data, stored in a file resource and compute summary
opal.assign.resource(o, "res", "RSRC.gps_data")
opal.assign.script(o, "gps", quote(as.data.frame(res)))
opal.execute(o, "summary(gps)")

# disconnect
opal.logout(o)
```

Step 5 - Troubleshooting

If anything goes wrong in an R server, check the output of the R commands in the R server logs.

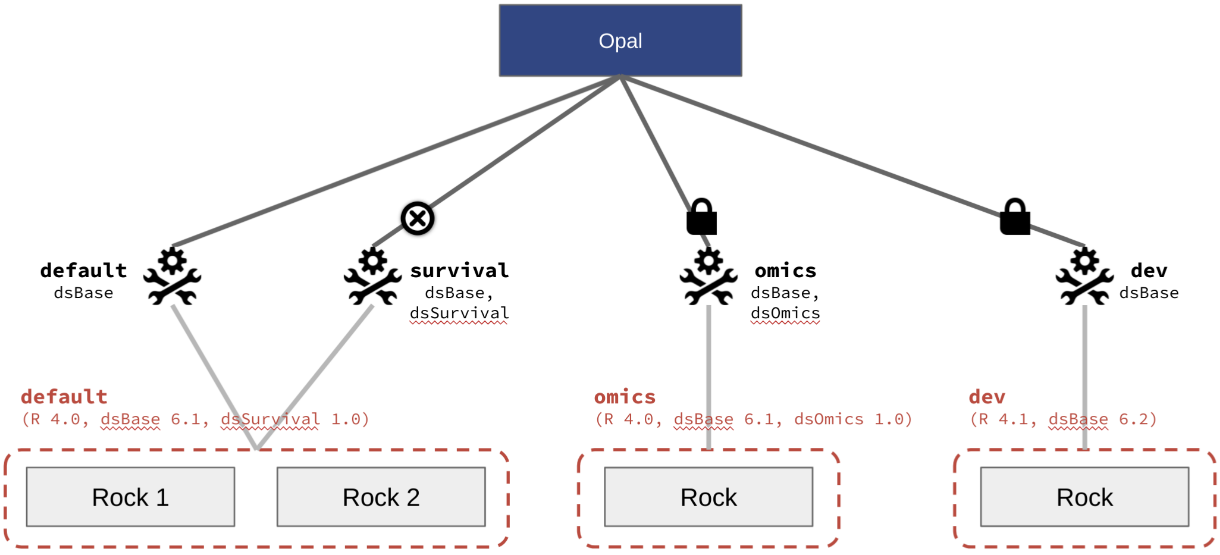
Note:

1. Go to the **Administration > R** page,
2. Select the R server profile/cluster,
3. Download the R server logs.

This log may contain information, when for instance an R package is broken or some R function calls output some error/warning messages without failing properly.

6.2.2 How to Set up DataSHIELD Server Profiles

Opal is able to connect to multiple R servers: see *R Server* documentation. R servers are organized in clusters. The cluster name is the R server profile name. From each R server profile, a DataSHIELD profile can be derived.



```
Different DataSHIELD settings associated to a cluster of R servers
Controls: enabled/disabled and permission to access
```

See also *How to Set up R/DataSHIELD Client Profiles*.

The following instructions will guide you in setting up the DataSHIELD profiles.

Step 1 - Setup R Servers

Everything starts with Opal connecting to different R servers, then follow the *How to Set up R Server Profiles* instructions.

Step 2 - Initialize DataSHIELD Settings

Connecting Opal to a R server with DataSHIELD R packages is not enough for having a functional DataSHIELD profile: DataSHIELD settings must be initialized, i.e. the allowed aggregate and assign functions along with R options must be declared. The following operations consist of asking Opal to inspect the R packages installed in the R server, to find their DataSHIELD statement.

From Administration Page

Note:

1. Go to **Administration > DataSHIELD** page
2. Select a profile in the *Profiles* section
3. Press **Initialize** in the profile's *Settings* section
4. Press **Enable** in the profile's *Status* section

The DataSHIELD settings are read from the installed DataSHIELD R packages and set as the profile's configuration.

Using R

The opalr R package (see *Using R* documentation) has many functions for DataSHIELD administration, their name starting with `dsadmin.`. Initializing and enabling a profile can be done from R as follow:

```
library(opalr)
o <- opal.login(username = "administrator", password = "password", url = "https://opal-demo.obiba.org")

# init and enable the 'default' DS profile
dsadmin.profile_init(o, "default")
dsadmin.profile_enable(o, "default")

opal.logout(o)
```

Step 3 - [optional] Add DataSHIELD Alternate Profiles

In Step 2, the main DataSHIELD profile was based on a R profile and the discovered DataSHIELD settings. It is also possible to declare a DataSHIELD profile based on the same R profile but with a different name and different settings (functions and R options).

From Administration Page

Note:

1. Go to **Administration > DataSHIELD** page
2. Press **Add Profile** in the *Profiles* section
3. Select the name of the R servers cluster on which the profile will be based (i.e. where the R sessions will be created), and *Save*
4. From the created profile, in the *Settings* section, you can *Initialize* the settings and amend the list of R functions exposed and the R option values
5. Press **Enable** in the profile's *Status* section

The DataSHIELD settings are read from the installed DataSHIELD R packages and set as the profile's configuration.

Using R

Create a DataSHIELD profile and modify its default settings as follows:

```
library(opalr)
o <- opal.login(username = "administrator", password = "password", url = "https://opal-demo.obiba.org")

# init and enable the 'geo-alt' DS profile with a different privacy level option
dsadmin.profile_create(o, "geo-alt", cluster = "geo")
dsadmin.profile_init(o, "geo-alt")
dsadmin.set_option(o, "datashield.privacyLevel", "10", profile = "geo-alt")
dsadmin.profile_enable(o, "geo-alt")

opal.logout(o)
```

Step 4 - [optional] Restrict Access to DataSHIELD Profiles

By default a DataSHIELD profile can be used by any DataSHIELD user. Restricting access to advanced users can be useful when:

- Privacy levels are more relaxed
- Some server-side DataSHIELD functions are still experimental
- A DataSHIELD package is being developed

From Administration Page

Note:

1. Go to **Administration > DataSHIELD** page
2. Select a profile in the *Profiles* section
3. Press **Restrict Access** in the profile's *Permissions* section
4. Press **Add Permission** and specify either user or group names who will be allowed to use this profile

Using R

Following the example in Step 3, call the appropriate `dsadmin.profile` function:

```
# only members of the 'developers' group can use this profile
dsadmin.profile_perm_add(o, "geo-alt", "developers", type = "group")
```

Step 5 - Use DataSHIELD Profiles from R Client

For the prerequisites, see the *Using DataSHIELD* documentation.

When setting up the DataSHIELD connection information, the DataSHIELD profile can be specified (otherwise it fallbacks to `default`). As an example:

```
library(DSOpal)
library(dsBaseClient)
builder <- DSI::newDSLoginBuilder()
# connect to 'study1' on its 'geo' profile
builder$append(server = "study1",  url = "https://opal-demo.obiba.org",
           user = "dsuser", password = "password",
           profile = "geo")
logindata <- builder$build()
conns <- DSI::datashield.login(logins = logindata)

# list available (and current) DS profiles
datashield.profiles(conns)

# list the allowed aggregate functions
datashield.method_status(conns, type = "aggregate")

# assign data and compute
datashield.assign.table(conns, "D", "CNSIM.CNSIM1")
ds.colnames("D")

datashield.logout(conns)
```

Note that:

- The profile name can be different from one DataSHIELD server to another.
- There is no check whether each DataSHIELD servers have the same profile settings.

For user-friendliness and reproducibility, it should be encouraged to coordinate when planning DataSHIELD profiles. Sharing the same R server Docker image and initialization R scripts should help.

When dealing with different DataSHIELD server profiles that require incompatible client-side packages, see proposed solution of *How to Set up R/DataSHIELD Client Profiles*.

6.2.3 How to Set up R/DataSHIELD Client Profiles

Following the possibility to have different R/DataSHIELD server profiles (see *How to Set up R Server Profiles* and *How to Set up DataSHIELD Server Profiles*), it may be necessary to have different R environment on the client side as well. This is mandatory in the case of DataSHIELD where a server-side package only works with a specific client-side package version.

This concept of R environment is covered by the renv R package which intends to provide **isolated**, **portable** and **reproducible** R projects. Using `renv` gives full control of the user on its R exection environment, without depending on an IT infrastructure (different interfaces for different profiles).

In order to help with setting up such project, you can start with the DSProjectTemplate.

Adding a Dependency on an R Package from GitHub

As an example some of the DataSHIELD R packages are not available in the official CRAN repository. For R packages saved on GitHub you can use the following procedure to include a new dependency:

Note:

1. Run `renv::install()` to retrieve a specific version of a R package from GitHub; for example `renv::install("neelsoumya/dsSurvivalClient@v1.0.0")` to install a released version of dsSurvivalClient
2. Edit your analysis script to make use of this package; for example `library(dsSurvivalClient)`
3. Run `renv::snapshot()` to register this package as part of your project environment

Removing a Dependency on an R Package

This is as simple as:

Note:

1. Edit your analysis script and remove the `library()` load statement of the package
2. Run `renv::snapshot()` to unregister this package from your project environment

6.2.4 How to Manage Authentication and Authorizations

Apply DataSHIELD Permissions

It is recommended to grant permissions by group, and then assign each user to the appropriate groups.

There are two kinds of permissions that need to be granted:

- Permission to use the DataSHIELD service, defined globally
- Permission to use data (table or resource) in the context of DataSHIELD, defined per table of per resource

Note that the DataSHIELD-compatible permissions (view dictionary and summary, without individual values access) are to be applied per table, whereas it is possible to apply similar permission (view, without resource credentials access) to any resource of a project in a single operation.

From Web Page

Note: As a system administrator:

1. [optional] Group local users (if not already done): go to **Administration > Users and Groups** and edit users so that they belong to the relevant group
2. [optional] Configure *Identity Providers* so that the users are automatically granted the appropriate group: go to **Administration > Identity Providers** and configure each ID provider (groups can be automatically applied or extracted from the received UserInfo).

3. Grant group the permission to *use* the DataSHIELD service: go to **Administration > DataSHIELD**, *Permissions* section (bottom of the page)

As a system administrator or as a project administrator:

- Grant group the permission to *view dictionnary and summary (no access to individual values)* of a Table: go to project's table page, select the **Permissions** tab
- Grant group the permission to *view (no credentials)* a Resource: go to the project's resource page, select the **Permissions** tab
- Grant group the permission to *view (no credentials)* any Resources of a project: go to the project's resources list page, select the **Permissions** tab

Using R

Use the following *opalr* R functions to manage permissions:

- Table permissions functions `opal.table_perm*` (opal.table_perm_add() etc.)
- Resource permissions functions `opal.resource_perm` (opal.resource_perm_add() etc.) and `opal.resources_perm` (opal.resources_perm_add() etc.)

The following example grants DataSHIELD permissions to `projectA` and `projectB` groups:

```
# load opal library
library(opalr)
# connect to the opal server
o <- opal.login(username = "administrator", password = 'password', url = 'https://opal-
↪demo.obiba.org')

# create a user in a group, call output is the generated password
pwd <- oadmin.user_add(o, "foo", groups = c("projectA"))

# use DataSHIELD service
dsadmin.perm_add(o, subject = c("projectA", "projectB"), type = "group", permission =
↪"use")

# single table
opal.table_perm_add(o, 'CNSIM', 'CNSIM1', c('projectA', 'projectB'), type = 'group',␣
↪permission = 'view')

# each table of a project
lapply(opal.tables(o, "CNSIM")$name, function(table) {
  opal.table_perm_add(o, "CNSIM", table, subject = c("projectA", "projectB"), type =
↪"group", permission = "view")
})

# single resource
opal.resource_perm_add(o, "RSRC", "CNSIM1", subject = c("projectA", "projectB"), type =
↪"group", permission = "view")

# any resources of a project
opal.resources_perm_add(o, "RSRC", subject = c("projectA", "projectB"), type = "group",␣
↪permission = "view")
```

(continues on next page)

(continued from previous page)

```
opal.lgout(o)
```

Use Personal Access Token

As documented in the *Two-factor Authentication* documentation, the PAT is the recommended authentication strategy, as it has a limited scope (project access and operations), and is revocable. Its usage is mandatory when the user is authenticated externally (see *Identity Providers* documentation).

Create a PAT

From Profile Page

Note:

1. Go to **My Profile** page (press the username on the top right corner)
2. Press **Add Access Token** and select a prepared token configuration (*DataSHIELD*, etc.) or a custom one
3. Fill in the token form
4. Copy the generated token and *Save*

Using R

Use the `opal.token*` functions to manage your PATs. More specifically, use the prepared token configurations opal.token_r_create(), opal.token_datashield_create(), or opal.token_sql_create().

```
# load opal library
library(opalr)
# connect to the opal server
o <- opal.login()

# the output of the call is the token
token <- opal.token_datashield_create(o, "test")

opal.logout(o)
```

Use the PAT

Replace in your R/DataSHIELD, or Python, scripts the username/password credentials by the *token* parameter.

In R

```
# load opal library
library(opalr)
# connect to the opal server with a token
o <- opal.login(token = "xxxxxxx", url = "https://opal.example.org")

# ...
```

In DataSHIELD

```
library(DSOpal)
library(dsBaseClient)
builder <- DSI::newDSLoginBuilder()
# connect to 'study1' with a token
builder$append(server = "study1",  url = "https://opal-demo.obiba.org",
               token = "xxxxxxxx")
logindata <- builder$build()
conns <- DSI::datashield.login(logins = logindata)

# ...
```

DataSHIELD and Central Authentication Service

In the DataSHIELD context, managing users is usually a pain for the infrastructure coordinator as each data node custodian must create a user, which takes time, and with potentially as many different passwords to keep safe.

An different setup is to use a Central Authentication Service (CAS), where users are registered once and properly configured (profiles/groups). Then each DataSHIELD Opal would connect to the CAS. Opal supports external *Identity Providers* using the standard OpenID Connect protocol.

The *opalr* R package does not currently support the OpenID dance (and it is anyway not appropriate for a scripting usage), then a user must login the Opal web interface of each node once, so that its user profile is validated and to create *Two-factor Authentication* that will be used in its DataSHIELD R scripts.

6.2.5 How to Improve Performance

When it comes to importing data and performing analysis in the R server(s), the performance of the whole system setup (Opal, database, R server) must match the user needs. The performance will depend on:

- the volume of data and its dimensions (lot of records or lot of variables or both)
- the number of concurrent users
- the type of analyses that are conducted
- the type of hardware hosting the system (virtual vs. barebone, shared vs. dedicated)
- the version of the software

The following are recommendations and possible solutions (that can be combined) for improving the performance of your system.

Use Resources

If data are too large to be imported in the Opal's database, consider using the *Resources* as described in *How to NOT Import Data*. This will also save CPU and memory usage at data assignment time in R.

Remove Unsused R Objects

Accumulating temporary R objects in the server-side R session will result in an increased usage of the R server's memory, with a potential of server failure. Use the DSI::datashield.rm() function, or the more informative dsBaseClient::ds.rm() function to release some R server memory.

Save your R/DataSHIELD Workspace

When conducting a R/DataSHIELD analysis, it can take some computation time to get intermediate results. Instead of redoing the computations again and again at each R/DataSHIELD session, it is possible to save a snapshot of your R server session into a file that will be kept safe in Opal. The following R/DataSHIELD sessions can then reinstate a previously saved workspace.

In the context of DataSHIELD, assign some data, make some preliminary computations and save R workspace:

```
library(DSOpal)
library(dsBaseClient)
builder <- DSI::newDSLoginBuilder()
builder$append(server = "study1",
           user = "dsuser", password = "password",
           url = "https://opal-demo.obiba.org")
builder$append(server = "study2",
           user = "dsuser", password = "password",
           url = "https://opal-demo.obiba.org")
logindata <- builder$build()

##
# start session: connect to the studies, no assignment
##
conns <- datashield.login(logindata)

# assign a table per study
datashield.assign.table(conns, "cnsim", list(study1 = "CNSIM.CNSIM1", study2 = "CNSIM.
↪CNSIM2"))
# assign a resource and coerce to a data.frame
datashield.assign.resource(conns, "cnsimClient", list(study1 = "RSRC.CNSIM1", study2 =
↪"RSRC.CNSIM2"))
datashield.assign.expr(conns, "cnsim.rsrc", quote(as.resource.data.frame(cnsimClient,␣
↪strict = TRUE)))

# get some summaries
ds.summary("cnsim")
ds.summary("cnsim.rsrc")

# remove the resource client symbol
ds.rm("cnsimClient")
# verify symbols
```

(continues on next page)

(continued from previous page)

```
datashield.symbols(conns)

# save remote R workspaces with a name
datashield.workspace_save(conns, "assigned")

# end session
datashield.logout(conns)
```

Then in the next DataSHIELD session, reinstate the saved R workspace:

```
library(DSOpal)
library(dsBaseClient)
builder <- DSI::newDSLoginBuilder()
builder$append(server = "study1",
           user = "dsuser", password = "password",
           url = "https://opal-demo.obiba.org")
builder$append(server = "study2",
           user = "dsuser", password = "password",
           url = "https://opal-demo.obiba.org")
logindata <- builder$build()

##
# start session: restore saved workspace
##
conns <- datashield.login(logindata, restore = "assigned")

# verify symbols
datashield.symbols(conns)

# get some summaries
ds.summary("cnsim")
ds.summary("cnsim.rsrc")

# end session
datashield.logout(conns)
```

Increase Hardware Power

Extracting data from the database requires memory and CPU. Assigning data consumes the hardware resources for the Opal and R applications. Analysing data in R consumes CPU (one core per R session) and memory. When installed on the same server (not recommended), the database, Opal and R applications are then running concurrently which can lead to a freeze of the system (shared CPU cores, memory swapping).

The solution can be:

- Either to increase the host's hardware power (better and more CPUs, more memory),
- Or to isolate each application in its own server host. This can be done easily and dynamically for the R server (see *Apps*).

Use Barebone Machine

Cloud facilities offer the possibility to start Virtual Machines (VM), which is convenient but not optimal for intensive computations. Several VMs are running on a single hardware server, then sharing the access to the CPU cores and to the memory. Even when these hardware resources are reserved, there is still the virtualization extra layer that can affect the computation performance.

The solution can be to use barebone servers, i.e. dedicated servers that are accessed directly and not shared with others.

Do R Server Load Balancing

When many users are using the same R server simultaneously (R server can handle multiple R sessions in parallel), all the system's CPU cores may be in use. In addition to that the sytem's memory is shared among all the concurrent R sessions, which can break the R sessions running out of memory when one of them is too greedy.

The solution can be to increase the hardware resources (see previous section) or to add more R servers. Opal supports connection with multiple R servers, both for defining R/DataSHIELD profiles (see *How to Set up R Server Profiles*) and for balancing the computation load. To achieve that:

- Several R servers must belong to the same `cluster`: see Rock's Cluster Node Configuration, corresponding to the ROCK_CLUSTER Docker environment parameter.
- These R servers must run on different hosts

Opal will take in charge the creation of the R sessions in the way that it optimizes the usage of the cluster of R servers.

The advantage of this solution is that it does not require to modify the setup (database and Opal). Only new R servers will have to be declared in Opal and this can be done dynamically (service discovery or self-registration as documented in *Apps*).

Keep Your Software Updated

New software version can bring new functionalities, and can also improve the operations performance. Unless stated in the release note and in the the release announcement, new versions of Opal are backward compatible. Make sure your IT department is keeping applications updated, which is also recommended for preventing security issues.

CHAPTER SEVEN

WEB INTRODUCTION

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

7.1 Requirements

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

CHAPTER

EIGHT

PROJECTS

8.1 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 (deprecated)				x
Tabular SQL Datasource		x	x	x
Limesurvey Datasource		x		
Opal Datasource		x		
MongoDB Datasource				x

8.1.1 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.

8.1.2 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 (deprecated)

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 (deprecated)* 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.

8.1.3 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 and is the recommended one for replacing deprecated Opal SQL datasources.

8.1.4 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.

8.2 Project Tables

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

See also details about how to manage tables and variables individually:

8.2.1 Table

A table can be a raw table or a view

Dictionary

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

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

Summary

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

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

Values

The values of the table can be seen in this section. the view port 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.

Permissions

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

View dictionary and summaries Permission

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

View dictionary and 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 and view summaries 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 dictionary and view 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 dictionary and view values 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.

Administrate Permission

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

Operations

Reconnect Resource

When the table is a view over a resource (see *Add View* in Resource documentation), it is possible to reset the connection with the underlying resource. This can be convenient when the connection is lost (because resource lives in a remote server for instance) or has failed unexpectedly.

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 an existing one or one that would be created for that purpose). Options are:

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

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

Export Variable Dictionary

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

View specific Operations

Download View XML

Only available if the table is a view.

Edit View

View over Tables

When the view is based on other tables, you can 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.

View over Resource

When the view is based on a resource (see *Add View* in Resource documentation), you can edit the views properties: table name, ID column name, resource reference etc. Depending on the type of operation, the connection with the underlying resource could be reestablished.

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.

8.2.2 Variable

A variable describes the data.

Operations

Add variable to View

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

Categorize this variable to another

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

Categorize another variable to this

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

Custom derivation

Derive a variable by editing its derivation script manually.

Remove

Removes the variable from the table.

Properties

This section displays the proerties of the variable:

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

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.

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.

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

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 version control system.

Commit Differences

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

Reverting Changes

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

Review Commit Differences

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

Values

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

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.

8.2.3 SQL

SQL is a very powerful language for data manipulation. SQL support in Opal allows to easily join tables, filtering, sorting and grouping data, make aggregations etc. SQL queries can be executed on one or more tables (or views) of a project from the web interface (or all projects using the programmatic API). Permission to access the values of the considered tables is required.

The supported SQL syntax is the the one of SQLite. More specifically see the SQL syntax and functions documentation.

The result of the SQL query can be downloaded from the web interface in CSV format. For a programmatic access to the SQL API, see the *SQL* python command and the `opal.sql()` function in the opalr R package.

Note that in Opal, there is no variable for accessing the identifiers. Then when performing assignment of the table data into the SQL environment, an identifiers column is added and called `_id` by default.

Note:

- **SQL API is read-only**. Statements like CREATE, ALTER, DROP, INSERT, DELETE are not supported.
- **Maximum number of columns is 2,000**. SQL queries cannot be executed on tables with more than 1,999 variables (+ identifier). Use views to extract the variables of interest.

Table and Variable Naming

When executed in the context of a project, the simple table name can be used:

```
SELECT * FROM CNSIM1 LIMIT 10
```

If this simple table name contains a `.` character it must be escaped by backquotes:

```
SELECT * FROM `StandingHeight.Baseline` LIMIT 10
```

When there is no project context, or when referring a table that is not in the current project, use the fully qualified table name with backquotes:

```
SELECT * FROM `CNSIM.CNSIM1` LIMIT 10
```

To desambiguate the column names, the table name can be used in `SELECT`, `JOIN` etc. statements:

```
SELECT CNSIM1._id, CNSIM1.GENDER, `CNSIM.CNSIM2`.PM_BMI_CATEGORICAL
  FROM CNSIM1
  LEFT JOIN `CNSIM.CNSIM2` ON CNSIM1._id = `CNSIM.CNSIM2`._id
```

The same escape rule applies to variable names, when they contain a `.` character:

```
SELECT `InstrumentRun.timeStart`
  FROM StandingHeight
  LIMIT 10
```

or with fully qualified table name:

```
SELECT `baseline.StandingHeight`.`InstrumentRun.timeStart`
  FROM `baseline.StandingHeight`
  LIMIT 10
```

Functions

Agregate Functions

See the aggregation functions documentation.

```
SELECT avg(LAB_HDL) as HDL_AVG, GENDER
    FROM CNSIM1
    WHERE LAB_HDL is not null
    GROUP BY GENDER
```

Date and Time Functions

See the date and time functions documentation.

```
SELECT *, date('now') AS extraction_date
    FROM CNSIM1
    LIMIT 10
```

Scalar Functions

See the scalar functions documentation.

```
SELECT round(LAB_HDL, 1) as HDL_ABS, GENDER
    FROM CNSIM1
    LIMIT 10
```

Paging

The directive `LIMIT` and `OFFSET` (combined with `ORDER BY`) can be used to extract some part of the data. The following query gets the 101st to 111th lines of the query result output ordered by the identifiers:

```
SELECT *
  FROM CNSIM1
  ORDER BY _id
  LIMIT 10 OFFSET 100
```

Union of Tables

When tables have the same columns, they can be stacked as follow:

```
SELECT * FROM CNSIM1
  UNION ALL SELECT * FROM CNSIM2
  LIMIT 10
```

Join Tables

Let’s join the tables `samples` (columns `_id`, `Donor`, `ConsentStatus`) and `donors` (columns `_id`, `Gender`):

```
SELECT Donor AS ID, samples._id AS sample_id, ConsentStatus, Gender
    FROM samples
    LEFT JOIN donors ON donors._id = samples.Donor
    LIMIT 10
```

And then make aggregations, for instance counting the number of donors having at least one “unknown” consent, per gender:

```
SELECT count(DISTINCT Donor) AS DonorsCount, Gender
    FROM samples
    LEFT JOIN donors ON donors._id = samples.Donor
    WHERE ConsentStatus LIKE "%unknown"
    GROUP BY Gender
```

8.2.4 Permissions

Specify the access rights to any table of the project and its content.

View dictionary and values of all tables Permission

Allow the user to see the data dictionary with individual-level data of any table.

Add table Permission

Allow the addition of a table or of a view, directly or via import or copy tasks.

Administrate Permission

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

8.2.5 Operations

Search Variables

Variables of the project’s tables can be searched. See *Search Variables*.

Download Dictionary

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

Backup Views

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

Import Data

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

The importation process follows several steps:

- Data format selection: file-based (CSV, Opal Archive), 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 described as follow:

Option	Description
Incremental	Opal is able to detect new or updated data, relying on entity identifier and some timestamps. By default the data import is not incremental, i.e. already existing data will be overridden.
Limit	A maximum number of data rows to be imported. Combined with the option, this allow to import small chunks of data at 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 corresponding system identifier for the selected mapping • Ignore unknown identifiers at import: only data with identifier having a corresponding system identifier in the selected mapping will be imported • Generate a system identifier for each unknown imported identifiers: a system identifier will be generated for each unknown imported identifier If no identifiers mapping is selected, the participant identifiers are imported as-is. Unknown participant identifiers will be automatically added in Opal.

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 must 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 only when one table is selected for copy).

Add Table

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

Add/Update table from Dictionary

A table can be added or updated from a data dictionary file. This data dictionary can be an Excel file (see Excel file template) or a view XML file (this can be obtained from an existing view by selecting "Download View XML"). For importing data dictionary from another format (SPSS file for instance), an alternate solution is to follow the process of importing data with the setting of limiting to 0 data rows (3rd screen in the import data wizard).

An advanced option offers the possibility to merge the data dictionaries when doing an update (otherwise default behavior is to override the properties and attributes).

Add View

This operation follows a step-by-step procedure:

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

Derived variable algorithms are expressed using Magma Javascript API.

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 data.

8.3 Project Resources

Some reference to *Resources* can be defined within a project. The resource types are discovered by scanning the R packages in the R server (restarting the R server triggers a new package scan) and find the ones implementing the Resource Forms API. Then extending Opal to new type of resources is just a matter of installing the appropriate R package (see *Packages* documentation).

See also documentation of resource reference details:

8.3.1 Resource

Details of a reference to a resource can be displayed. Users not having the permission to see the credentials part of a resource will still be able to see its location. The resource page also displays information about the R package providing the resource type.

Permissions

Specify the access rights to a single resource.

View resource Permission

View resource without having access to the associated credentials (DataSHIELD compliant permission).

Administrate Permission

Allow all operations on the resource (including removal).

Operations

Add View

A view can be created over a resource. This operation is relevant only if the resource can be coerced to a tabular representation. Having a view over a resource allows to give a visibility to the data dictionary and the summary statistics of the resource, for the purpose of data documentation (in Mica for instance).

This view will establish a connection with a resource object in a background R server session. If the resource connection requires a specific R package that is installed only in a specific R server, Opal will try to guess the most appropriate R server profile or the R server profile name can be provided. The variables of the view will be initialized with the observed columns of the tibble's representation of the resource. The identifiers will be extracted from the first column, or from the one specified in the view creation form. The values will be queried from the underlying tibble object.

After the resource's view has been created, it is possible to rename variables, change their value type, manage categories and annotate them with taxonomy terms. It is not possible to make more complex derived variables, such as advanced data transformation or the combination with other column values.

Test Resource

Testing a resource will consist of assigning the resource reference into a R server session and verify that the resource type is correctly identified, i.e. the assigned R object is of class *ResourceClient* as defined by the resourcer R package. Note that this does not prove that the underlying data are accessible as getting these could be a deferred R operation.

Edit Resource

Edit the current resource, anything can be changed except the name.

Duplicate Resource

Duplicate a resource (type, parameters and credentials), useful when several resources are defined in the same data management system (several SQL tables in the same database for instance). Also useful for changing the resource name.

Remove Resource

Remove a resource reference (the underlying data management system is not affected).

8.3.2 Permissions

Specify the access rights to any resource of the project.

View any resource Permission

View any resource without having access to the associated credentials (DataSHIELD compliant permission).

Administrate Permission

Allow all operations on any resource of the project (including addition and removal).

8.3.3 Operations

Add Resource

To perform a resource addition the user is proposed to select the resource category and then select the appropriate resource type. Note that the same resource type can belong to several categories: these categories usually represents the data format, the way to access the resource or the domain of interest etc. The associated resource form will capture the different elements of the resource reference details (parameters and credentials), from which the R resource object will be built on R user request.

Edit Resource

The edition of a single resource can be done from the list of resource references. A resource reference cannot be renamed, instead use the *Duplicate Resource* operation to declare the same resource with a different name.

Remove Resource

Single or bulk removal of resource references (the underlying data management system is not affected).

8.4 Querying for Variables

The data dictionary (i.e. the variables) can be searched for. Each variable properties, 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.

8.4.1 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 <category>-<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

NINE

SEARCH

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

9.1 Search Variables

Search for variables using facets and full-text query.

9.1.1 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.

9.1.2 Full-text Search

The full-text search applies to:

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

Wildcard can be used.

9.1.3 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.

9.1.4 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.

9.2 Search Entity

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

9.2.1 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.

9.3 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.

9.3.1 Prerequisite

Only tables which values have been indexed can be searched.

9.3.2 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:

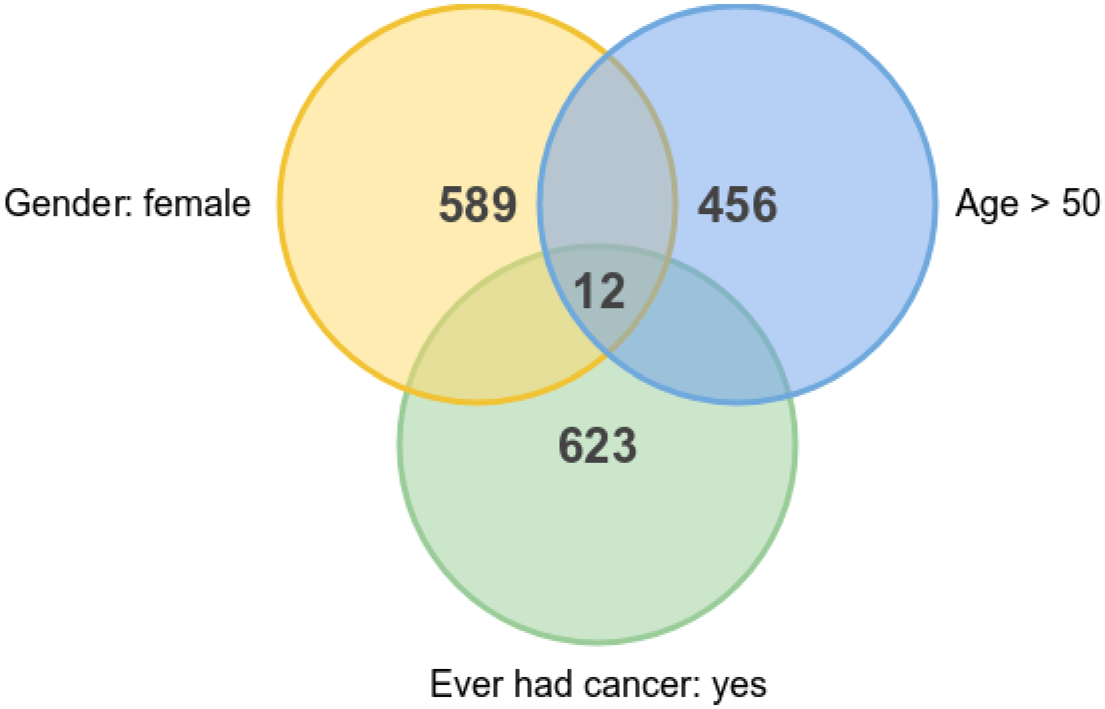
Operation	Description
All	The values of the variable can be any value (empty or not empty).
Empty	The values of the variable are empty (null value or value sequence with all null values).
Not empty	The values of the variable are not empty (not null value or value sequence with at least one not null value).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

9.3.3 Identifier Criterion

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

9.3.4 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

TEN

ADMINISTRATION

Opal manages databases for two distinct purposes: for holding the participants identifiers and for holding the variable catalog and participant data. Since Opal 3.0, it is not required to set up neither an identifiers database, nor a data database:

- If no identifiers database is defined, the *Identifiers Mappings* service when performing import/export will be disabled.
- If no data database is defined, no data can be imported in any project, only *Project Resources* can be defined and used for analysis.

10.1 General Settings

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

10.1.1 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.

10.1.2 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.

10.2 Identity Providers

10.2.1 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).

10.2.2 Personal Access Tokens

Because of the redirect of the user to the original OpenID Connect server login page, this authentication realm is not suitable for scripting/command line tools such as R, Python etc. The solution for a user defined in that kind of realm is to create one or more personal API access tokens (see *My Profile*) to perform authentication in scripting clients.

10.2.3 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:

- *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.
- *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.
- *Groups* are the group that are to be automatically applied to any users signing in through this ID provider.
- *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.
- *Username Mapping* is an optional field name in the UserInfo object which value will represent the Opal user name.
- *Groups Mapping by Claim value* is an optional field name in the UserInfo object (that is returned by the ID provider) that contains the group names to which the user belongs. These will be automatically applied to the user's profile. Such field is not one of the standard claims and needs to be explicitly set. The expected value type associated to this claim is either an array of strings, or a string which group names are separated by spaces (or commas). In the case of Agate, the groups claim name is `groups`.
- *Groups Mapping by Javascript* is an optional Javascript code chunk that will process the UserInfo object to extract a group name or an array of group names to which the Opal user will belong.

More advanced options are:

- *Nonce* to validate ID token with Nonce. Uncheck this option when Agate is used as a ID provider.
- *Connect Timeout* is the maximum time to wait before a connection is established, in milliseconds. Zero implies no timeout (default).
- *Read Timeout* is the maximum time to wait before a response is received, in milliseconds. Zero implies no timeout (default).
- *Public URL*, in case Opal is accessible from different public URLs, it is possible to specify which one is to be used for this ID provider. When not specified, the system's public URL applies. Note that this ID provider will be only visible when Opal base URL matches this public URL.

Note: To register Opal as a client of the OIDC provider it will be necessary to provide its callback URL which is: `https://opal.example.org/auth/callback/`.

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).

10.3 File System

Opal has its own file system accessible from the Dashboard, a project and the Files administration page.

10.3.1 File Browser

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

10.3.2 Operations

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

Add Folder

Adds a new sub-folder in the current location.

Upload

Uploads a file in the current location.

Download

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

Note: A zip file can be password protected to secure the downloaded data.

Delete

Deletes the selected files and folders from the current location.

SFTP

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

- Host: IP or the host name where the Opal server is running
- Protocol: SFTP
- Port: 8022 (In case of a firewall, this port must be accessible to the client)
- Logon Type: Normal (requires a valid username and password)

10.4 Users and Groups

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

10.4.1 Users

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

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

10.4.2 Groups

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

10.4.3 Operations

Add User with Password

The password must contain at least 8 characters, with at least one digit, one upper case alphabet, one lower case alphabet, one special character (which includes `@#$%^&+=!`) and no white space. After a user has been added, a personal folder is added in the home folder of the Opal's *File System*.

Add User with Certificate

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

Edit User

Change password or X.509 certificate or group membership.

Remove User

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

Disable User

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

Remove Group

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

10.5 R

R servers can be managed from this administration page. A R server is a Rock application.

10.5.1 R Servers

The R servers are grouped by cluster. Each cluster can be administrated individually. See the *R Server* documentation for understanding the underlying concepts of the integration of Opal with R.

Status

The R servers access can be tested, stopped and started. Stopping the R servers removes all the R sessions that could be in operation.

The output of the R commands that are executed on the R server are printed in the *Rserve.log* file that can be downloaded. This file is useful for troubleshooting (bad R command syntax, network failure, system dependency missing etc.).

Servers

Opal can connect to several R servers. All these servers are expected to be identical in terms of R base and packages versions. If this is not the case, this can be fixed by removing/installing packages: these operations apply to all the R servers in parallel.

Each R server can be started/stopped and log can be downloaded individually.

The R servers are provisioned as *Apps* and therefore will be managed as such, i.e. either discovered or self-registered.

Packages

The list of all the R packages installed on the R server(s) side is available. Note that some of the packages are installed at a system level location (such as */usr/local/lib/R/site-library*) whereas others are at a location that belongs to the *rserver* user (usually */var/lib/rserver/R/library*). Only the latter can be edited, i.e. packages can be removed or installed. R uses an order for the package lookup and the user's one (*rserver*) comes before the system ones. Then it is valid to have the same package installed at different locations and with different versions.

The CRAN repositories that are referred when performing install or update package operations are the ones defined in the system configuration `org.obiba.opal.r.repos` (see *R Server Configuration*).

Install single package

Several installation methods are proposed:

- standard installation using the *install.package()* function based on the configured CRAN repositories.
- remotes installation from a GitHub source repository, in which case the Git reference (commit number, tag or branch name) and the fully qualified package name (using the pattern *someUser/someRepo*) are mandatory.
- Bioconductor manager installation.

Update all packages

This operation calls the utils::update.packages() function to update all packages that can be updated to their latest version in the *rserver* R library location.

Remove single package

This will be effective if the package location is not a system one.

10.5.2 R Sessions

Each user R session is created and managed by Opal, whether the context is plain R, DataSHIELD or report execution. The list of the R sessions reports who owns the R session, when was the last R command executed (Last access) and whether a R command is in progress (Status). When removing a R session, the remote R work directory is destroyed and all the associated R resources are freed.

10.5.3 R Workspaces

Opal offers the possibility in the Opal R client API (see opalr) to save the image of the remote R session into a file (within any files that could be found in the R session working directory) in a safe location on Opal server for latter reinstate. This service is available for DataSHIELD sessions too. The archived workspaces can be removed at any time.

10.5.4 Permissions

The use of the R service requires permission. Needless to say that it should be granted to trusted users only as the R scripting capabilities are potentially harmful for the hosting system (Opal data are safe though as a user could not transfer a Opal table to a R data.frame in the R server if s/he has no permission to see the values of this table).

10.6 DataSHIELD

The DataSHIELD platform configuration can be managed from this administration page. See the *R Server* documentation for understanding the underlying concepts of the integration of Opal with R.

10.6.1 Packages

The DataSHIELD R packages management is done per R servers cluster.

The list of DataSHELD R packages (packages that declare some DataSHIELD configuration) is discovered by scanning the R packages installed on each R server of the cluster. This DataSHIELD configuration is expected to be either in the `DESCRIPTION` file or in a `DATASHIELD` file to be found in the package's installation directory. The items of this configuration are:

- **AggregateMethods** is a comma-separated list of key-value pairs, where the key is the R function name to be used on the R client-side and the value is the R function name that will be applied in the R server-side (if omitted, the server function name is assumed to be the same as the client function name within the R package namespace). These functions perform aggregation operation and return non-disclosive data.
- **AssignMethods** is a comma-separated list of key-value pairs, where the key is the R function name to be used on the R client-side and the value is the R function name that will be applied in the R server-side (if omitted, the server function name is assumed to be the same as the client function name within the R package namespace). These functions perform data assignment in the R server session and do not return values.

- **Options** is a comma-separated list of key-value pairs, where the key is the R option name and the value is the R option value, to be applied after the DataSHIELD R session creation.

Add Package

R packages are installed from the CRAN repositories defined in the Opal system which includes by default the OBiBa CRAN repository. See the Opal system configuration to modify this repository setting.

Adding "all DataSHIELD packages" means installing the datashield R package which is a meta-package with dependencies.

Adding a specific DataSHIELD package will install the R package from the CRAN repositories or from a GitHub repository.

When there are several R servers in the cluster, adding a R package will add it to all the R servers.

Delete all Packages

This will uninstall all the identified DataSHIELD R packages from the R server (as soon as their installation location is accessible!).

Remove Package

This will uninstall the selected DataSHIELD R package from the R server (as soon as its installation location is accessible!).

When there are several R servers in the cluster, removing a R package will remove it from all the R servers.

Publish

Read the DataSHIELD configuration declared in the R package (as described above) and merge it in the corresponding DataSHIELD profile (the one with the same name as the considered R server cluster).

Unpublish

Read the DataSHIELD configuration declared in the R package (as described above) and removes it from the corresponding DataSHIELD profile (the one with the same name as the considered R server cluster).

10.6.2 Profiles

A DataSHIELD profile is a combination of a R server profile (or cluster) with some specific settings. When end-user login in a DataSHIELD context and providing a profile name, the R session will be created in the corresponding R server cluster and function call filtering will be applied based on the DataSHIELD profile settings.

Status

A DataSHIELD profile can be disabled (recommended when modfying the settings). When disabled, no DataSHIELD R session can be created using this profile.

Permissions

By default a DataSHIELD profile can be used by any user with global DataSHIELD permission. It is also possible to restrict access to a profile by applying specific permissions. Note that granting permission to use a DataSHIELD profile also grants permission to use DataSHIELD generally.

Settings

DataSHIELD settings control the R operations that can be performed in the DataSHIELD R session.

Initialization

DataSHIELD settings can be initialized by selecting which DataSHIELD R packages methods and options are to be included in the allowed operations.

DataSHIELD methods are the function names that a DataSHIELD client is allowed to call. Each of these functions is mapped to a server-side function. This server-side function can be either a function name declared in its namespace (for instance `base::ls` or `dsBase::colnamesDS`) or a custom R function script (for advanced users).

Aggregate Methods

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

Assign Methods

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

Options

The list of R options to apply after creating a DataSHIELD R session. These options are used to alter the behavior of the server-side functions (control of the privacy threshold for instance).

10.6.3 Permissions

Use Permission

The use of the DataSHIELD service requires permission. This permission is to be combined with permissions on project tables and/or resources so that user can perform assignment operations in the R server.

Administrate Permission

Permission to install DataSHIELD R packages and to modify the global DataSHIELD configuration.

10.7 Taxonomies

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

10.7.1 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.

10.7.2 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.

10.8 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 (deprecated)	x		
MySQL, MariaDB, PostgreSQL	Tabular SQL	x	x	x
MySQL, MariaDB, PostgreSQL	Limesurvey		x	
MongoDB	Opal Documents	x		

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

Note that for more recent MySQL servers using the SHA2 authentication (that is known to not be working with JDBC), you should use the `mysql_native_password` directive when creating the user:

```
CREATE USER 'opal'@'localhost' WITH mysql_native_password IDENTIFIED BY '<opal-user-
↪password>';
```

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: [
      {
        "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*.

10.8.2 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 (deprecated)

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.

Deprecation notice: starting from Opal 5.0, the "Opal SQL" schema will not be supported. The project backup/restore service can be used for the migration (available from the project administration page or using the R/Pyhton libraries).

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.

10.8.3 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.

10.9 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

10.9.1 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.

10.9.2 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).

10.9.3 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).

10.9.4 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.

10.10 Apps

An application is an external service that can be used by the system. Applications can be discovered from their URL or can register themselves using the application authentication token.

10.10.1 Registry

The registered apps can be unregistered. Note that an app that was discovered, will be automatically registered again. To remove it completely it is necessary to remove it from the apps discovery configuration and then unregister it.

10.10.2 Configuration

For now, the only type of app known and supported is the Rock R server.

Self-registration

To register itself, an application must provide a registration token in its request. See *Apps Configuration* to set a default value for this token. Then periodical checks are performed: if the application cannot be reached or reports to be dysfunctional, it is automatically unregistered. With the self-registration mechanism it is easy to add more computation power to Opal without changing the configuration.

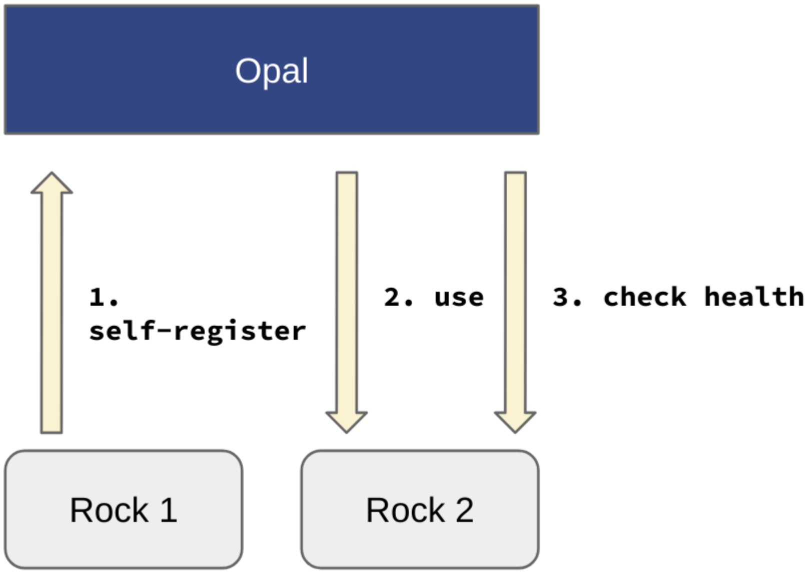


Fig. 1: Rock server self-registration.

Edit Registration Token

The apps registration token can be defined at system level (see *Apps Configuration*) and overridden from this administration page. Use the Generate button to create a secure token. Leave empty to reset to system's default.

Discovery

Applications can be discovered at runtime. When discovered, the application is automatically registered. Then periodical checks are performed: if the application cannot be reached or reports to be dysfunctional, it is automatically unregistered. See *Apps Configuration* for setting the discovery interval and the default hosts to lookup. With the discovery mechanism it is easy to set up some default external services.

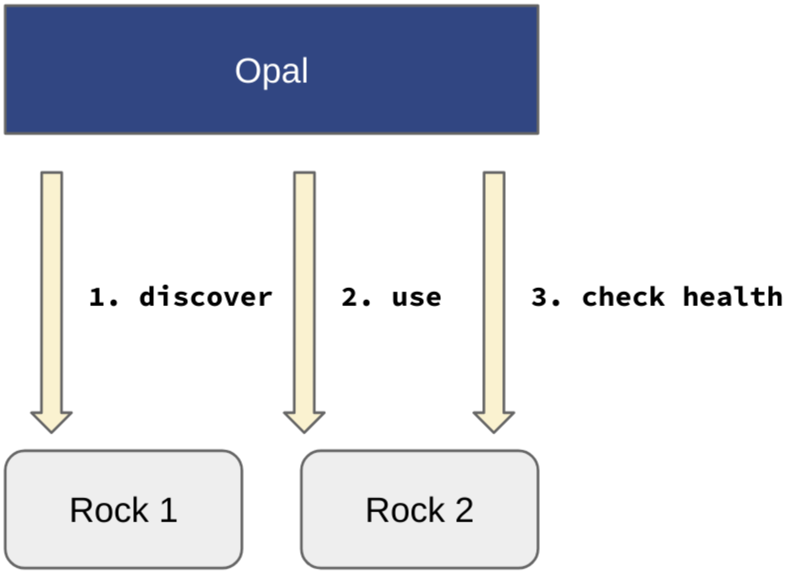


Fig. 2: Rock server discovery.

Add Rock App

Add a Rock R server to discover, by providing its URL (http…) and selecting an authentication strategy:

- **Default**, the credentials specified by the system configuration will apply (see *R Server Configuration*).
- **Administrator**, Opal will use a Rock administrator user both for managing the R server (start/stop server and install/remove packages) and creating R sessions.
- **Manager and User**, Opal will use a Rock manager user for managing the R server and a standard Rock user for creating R sessions.

See Rock Security documentation.

10.11 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

ELEVEN

MY PROFILE

Every user in Opal has a profile. This page is accessible by clicking on the user name on the top right corner of the web interface.

11.1 Account Settings

This section gives information about the user account: groups and how to change the password. The password must contain at least 8 characters, with at least one digit, one upper case alphabet, one lower case alphabet, one special character (which includes `@#$%^&+=!`) and no white space.

As Opal is able to delegate the user authentication to tier systems (such as Agate or *Identity Providers*), Opal may have no control on the user's password. Unless the user has been defined in Opal (the *opal-user-realm*), the password update may be delegated to the original realm of the user.

11.2 Two-factor Authentication

Two-factor authentication (2FA) is an extra step added to the log-in process: in addition to your username and password, a temporary code is requested. This temporary code is to be provided by an "authenticator" app installed on your mobile phone. The technology used is the time-based one-time password (TOTP), that consists of generating synchronized temporary codes both in the mobile app and the Opal server. There are several authenticator apps available in app stores; we recommend installing either **Microsoft Authenticator** (free, fully featured and robust solution, available on Android and Apple app stores) or **FreeOTP+** (open source solution, available on Android app store only).

Note that the 2FA feature is not available for users that identify from an external ID provider (i.e. through the OpenID Connect protocol, or in Agate). It is assumed that any 2FA/multi-factor auth process would be part of the external authentication flow.

The process of enabling 2FA is the following:

- Login with your username and password,
- Go to your profile page,
- Install an Authenticator app on your mobile phone (see above for recommended ones),
- Press "Enable 2FA": a QR code appears (shown only once!), to be scanned by your Authenticator app to register your account's 2FA settings.

To verify:

- Logout and login again with your username and password,
- Press "Sign In" and enter the requested temporary PIN code provided by the Authenticator app and "Validate".

In case the Authenticator app settings are lost, you can contact the system's administrator to disable your 2FA setting: as an administrator, go to Administration > Profiles pages and in the list of user profiles, press "Disable 2FA" for the considered user.

When 2FA is enabled, it affects the client libraries:

- **R**, when using *opal.login()* function with username and password you will be prompted to enter the PIN code.
- **Python**, the Opal Python commands accept the argument `--otp` (stands for "one-time password") to capture the PIN code from the prompt.
- **Java**, the PIN code cannot be provided.

Note that the 2FA mechanism does not apply when authenticating with a Personal Access Token. This "API key" is the recommended authentication process.

11.3 Personal Access Tokens

Personal access tokens can be created for use in scripts and on the command line (using R or Python client API). Be careful, these tokens are like passwords so you should guard them carefully. The advantage to using a token over putting your password into a script is that a token can be revoked, and you can generate lots of them. See also this Personal Access Token page.

In addition to that, the scope of the access granted to the token can be restricted by projects, data access and operations that can be performed on these projects and system services. Note that **the personal access token does not grant new permissions** but rather alter the ones you have.

The personal access token is also the only way to authenticate in a script for users defined in delegated *Identity Providers*, as the process of authentication for such realms implies the redirect of the user to a web page for manual login. The token is guaranteed to be safe for the Opal server as it is created by the user itself.

For these reasons, the personal API access token is the recommended way for authenticating within a Opal server (since Opal 2.15).

11.3.1 Settings

Projects

The projects that are accessible using the token can be limited. When none is enumerated, all the projects accessible by the user will be accessible using the token.

Project Data

The read/write operations can be controlled:

- **Default**, the user permissions apply,
- **Read only**, no data can be imported, nor modified/deleted. Individual-level data are still accessible.
- **Read only, without individual-level data**, no data can imported/exported, nor modified/deleted/extracted. Only reporting, analysis or DataSHIELD actions can be performed.

Project Tasks

This scope is for controlling which tasks can be launched on project:

- **Import**, import data (not available when project's data are read-only),
- **Export**, export data (not available when project's data are read-only without access to individual-level data),
- **Copy**, copy data (not available when project's data are read-only)
- **Backup**, backup project (not available when project's data are read-only without access to individual-level data),
- **Restore**, restore project (not available when project's data are read-only),
- **Report**, execute a report,
- **Analyse**, execute an analysis,
- **Import VCF**, import a VCF, when a VCF store plugin is installed (not available when project's data are read-only),
- **Export VCF**, export a VCF, when a VCF store plugin is installed (not available when project's data are read-only without access to individual-level data).

Project Administration

This scope of operations is for managing projects:

- **Create**, to create new projects, not available when project access is restricted to some enumerated ones (otherwise created project would not be accessible),
- **Update**, to update a project settings (does not apply to project's data),
- **Delete**, to delete a project.

Services

Along with project data, some system services can be used:

- **R**, which allows to create a plain R session in the R server backend, and assign some data (tables or resources), as soon as the user and token have permission to read individual-level data.
- **DataSHIELD**, which allows to create a DataSHIELD's R session in the R server backend, and assign some data (tables or resources), even when the user and token have not the permission to read individual-level data.
- **SQL**, which allows to make *SQL* queries on tables, as soon as the user and token have permission to read individual-level data.
- **Administrate system**, which allows to manage plugins, DataSHIELD configuration and much more (*administrator* users only).

11.3.2 Operations

Remove

You can permanently remove a token, effect is immediate.

Note that if the token has reached the end-of-life (system setting, by default there is no expiration timeout), it will be automatically removed, no action needed.

Renew

There is an inactivity timeout (system setting, 2 months by default) after which a token is not functional. When a token has been marked as being inactive, it can be renewed an unlimited number of times (until the token expires).

11.3.3 Examples

Example of usage in R (see section *Using R*):

```
o <- opal.login(token='dXvJKhk17RiO0TguRmR0EQlJxweCFyUX', url='https://opal-demo.obiba.
↪org')
...
```

Example of usage in Python (see section *Python Commands*)

```
opal dict "CNSIM.*" --opal https://opal-demo.obiba.org --token
↪'dXvJKhk17RiO0TguRmR0EQlJxweCFyUX'
```

Example of usage with cURL command line;

```
curl -H "X-Opal-Auth: dXvJKhk17RiO0TguRmR0EQlJxweCFyUX" -H "Accept: application/json" -X␣
↪GET https://opal-demo.obiba.org/ws/projects
```

11.4 Bookmarks

Bookmarks are shortcuts to specific pages (project, table, variable) that can be selected by clicking on the start icon on the right of the title.

CHAPTER

TWELVE

R DATASHIELD INTRODUCTION

12.1 What is R?

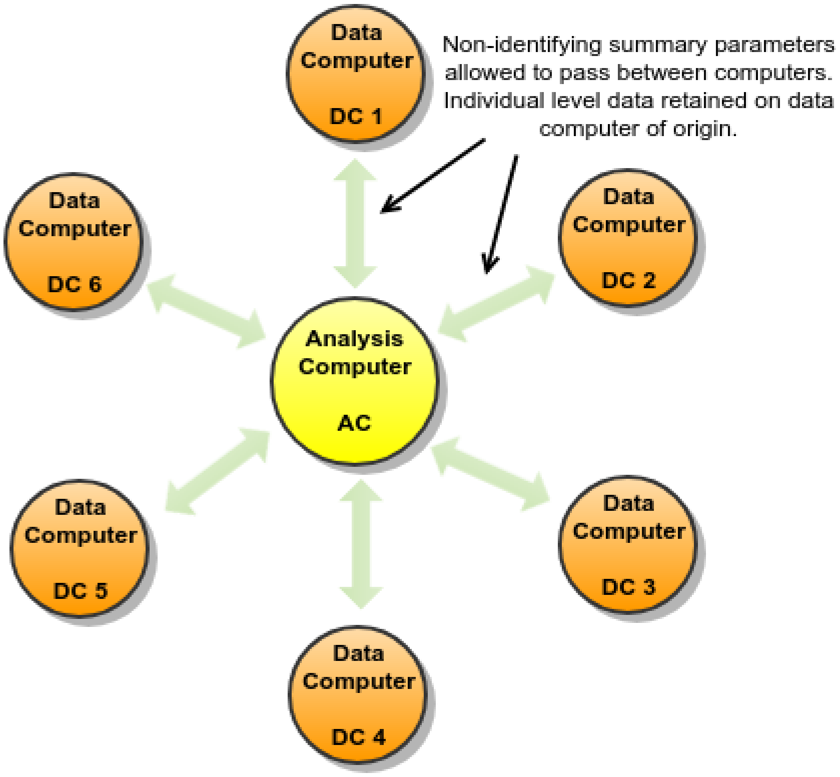
R is a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc.

Please consult the R project for further information.

12.2 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.

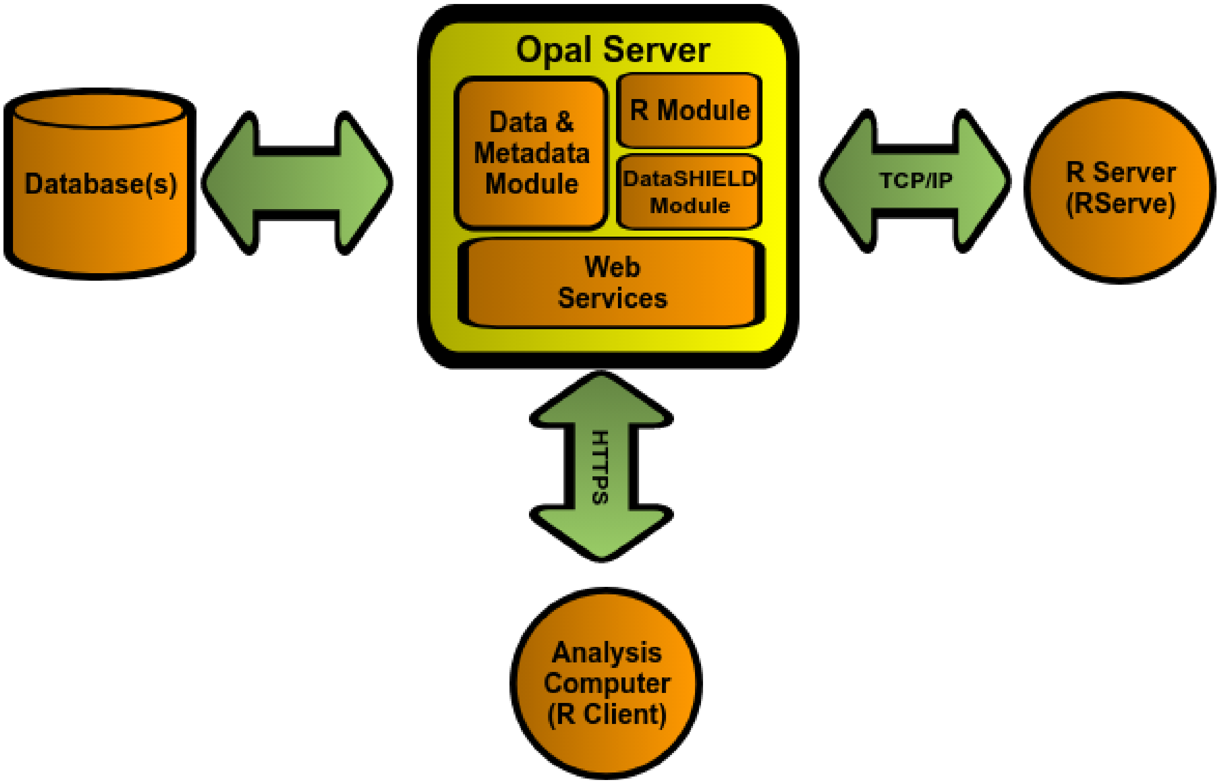

Data Computer DC 1
Non-identifying summary parameters allowed to pass between computers. Individual level data retained on data computer of origin.
Data Computer DC 6
Data Computer DC 2
Analysis Computer AC
Data Computer DC 5
Data Computer DC 3
Data Computer DC 4

Please consult the DataSHIELD project for further information.

12.3 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)



12.3.1 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.

12.3.2 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.

12.3.3 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.

12.3.4 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.

12.3.5 Web Services

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

12.3.6 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.

12.3.7 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

THIRTEEN

USING R

13.1 Prerequisites

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

13.2 Installation

The Opal Client R package is available in the official CRAN: see opalr CRAN page

You can install the Opal package and its dependencies with this command within an R session:

```
install.packages('opalr')
```

13.3 Usage

Accessing Opal data using R is straightforward:

```
# load opal library
library(opalr)

# get a reference to the opal server
o <- opal.login('administrator','password', url = 'https://opal-demo.obiba.org')

# assign some data to a data.frame
opal.assign.table(o,'CNSIM1','CNSIM.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)

# or send a R script to the R server and execute it
rval <- opal.execute.source(o, '/path/to/script.R')

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

See also Opal R documentation.

13.4 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)
```

The **recommended authentication method is by using a personal access token**: projects and operations can have a token's specific limited scope. See *Two-factor Authentication* documentation.

```
o <- opal.login(token='dXvJKhk17RiO0TguRmR0EQlJxweCFyUX', url='https://opal-demo.obiba.
↪org')
```

CHAPTER

FOURTEEN

USING DATASHIELD

14.1 Prerequisites

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

14.2 Installation

The Opal Client and DataSHIELD Interface R packages are available in the official CRAN:

- opalr CRAN page
- DSI CRAN page
- DSOpal CRAN page
- DSLite CRAN page

You can install the DataSHIELD implementation for Opal R packages and dependencies with this command within an R session:

```
install.packages('DSOpal')
```

14.3 Usage

14.3.1 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 Permissions section in *Project Tables* and in *Table* to know how to grant access to a table.

These access rights can be granted to a user or a group of users.

14.3.2 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.

14.3.3 DataSHIELD Usage

First thing required to use DataSHIELD is to load datashieldclient, the DataSHIELD base package for the client, into your R environment:

```
# Install dsBaseClient and dependencies if not already done
install.packages('dsBaseClient', repos=c(getOption('repos'), 'https://cran.obiba.org'))

# Load DataSHIELD base package
library(dsBaseClient)
```

14.3.4 Create a DATASHIELD Login Object

Every DataSHIELD operation requested on the analysis nodes require user authentication. The authentication credentials may be different in each of these server nodes. The first step is then build a DataSHIELD login data object that will group together the credentials and the connection details. The DSI package provides a utility R class to build this object.

```
# The login data object is a data.frame
builder <- DSI::newDSLoginBuilder()
builder$append(server="server1", url="https://opal.study1.org",
               user="dsuser", password="password")
builder$append(server="server2", url="https://opal.study2.org",
               token="123456789")
logindata <- builder$build()

# Then perform login in each server
library(DSOpal)
connections <- datashield.login(logins=logindata)
```

14.3.5 Invoking DataSHIELD Methods

Every DataSHIELD functions defined in the DSI package starts with *datashield.*. These functions allow to perform assignment and aggregation operations along with session management operations (workspaces, symbols, connection objects discovery etc.).

14.3.6 Working with Server-Side R

The following is based on the DSI functions that are fully described in the DSI README.

Assignments

First operation is usually a data assignment, either by directly assigning the table as a *data.frame* object or by assigning a resource as a *ResourceClient* object (as defined in the resourcer package).

```
# assign Opal tables to symbol D
datashield.assign.table(connections, symbol = "D",
                        table = list(server1 = "CNSIM.CNSIM1",
                                     server2 = "CNSIM.CNSIM2"))
# assign Opal resources to symbol rsrc
datashield.assign.resource(connections, symbol = "rsrc",
                           resource = list(server1 = "RSRC.CNSIM1",
                                           server2 = "RSRC.CNSIM2"))
```

It is possible to filter the variables of a table to be assigned:

```
# Assign some enumerated variables from 'opal-data.Table' to the TBL symbol as a data.frame
datashield.assign.table(connections, symbol = "D",
                        table = list(server1 = "CNSIM.CNSIM1",
                                     server2 = "CNSIM.CNSIM2"),
                        variables=list('LAB_GLUC','LAB_HDL'))
```

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(connections, 'some.data', quote(c(1:10)))
datashield.assign(connections, '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(connections)
# Remove a symbol from each Opal for the current datashield session
datashield.symbol_rm(connections, '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.

```
# Use the 'aggregate' method to invoke 'length'
# This form is used to invoke methods not defined by default
datashield.aggregate(connections, 'length(D$BMI)')
```

14.4 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.

14.4.1 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.

14.4.2 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.

14.4.3 Contributing to DataSHIELD Packages

DataSHIELD packages sources are hosted on GitHub. Some DataSHIELD developers documentation is also available. For more information visit the DataSHIELD web site.

CHAPTER

FIFTEEN

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).

15.1 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

15.2 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).

15.2.1 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.

15.2.2 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.

15.2.3 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.

15.2.4 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.

15.3 Advanced Examples

See advanced examples in the table folder, where opal-table.Rmd features:

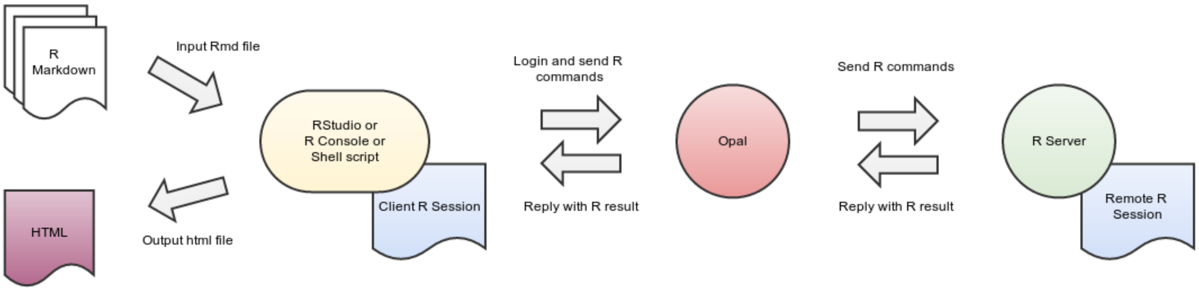
- R Markdown sub-reports,
- inline css-styling,
- access to Opal variables description.

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

15.4 Report Execution Flows

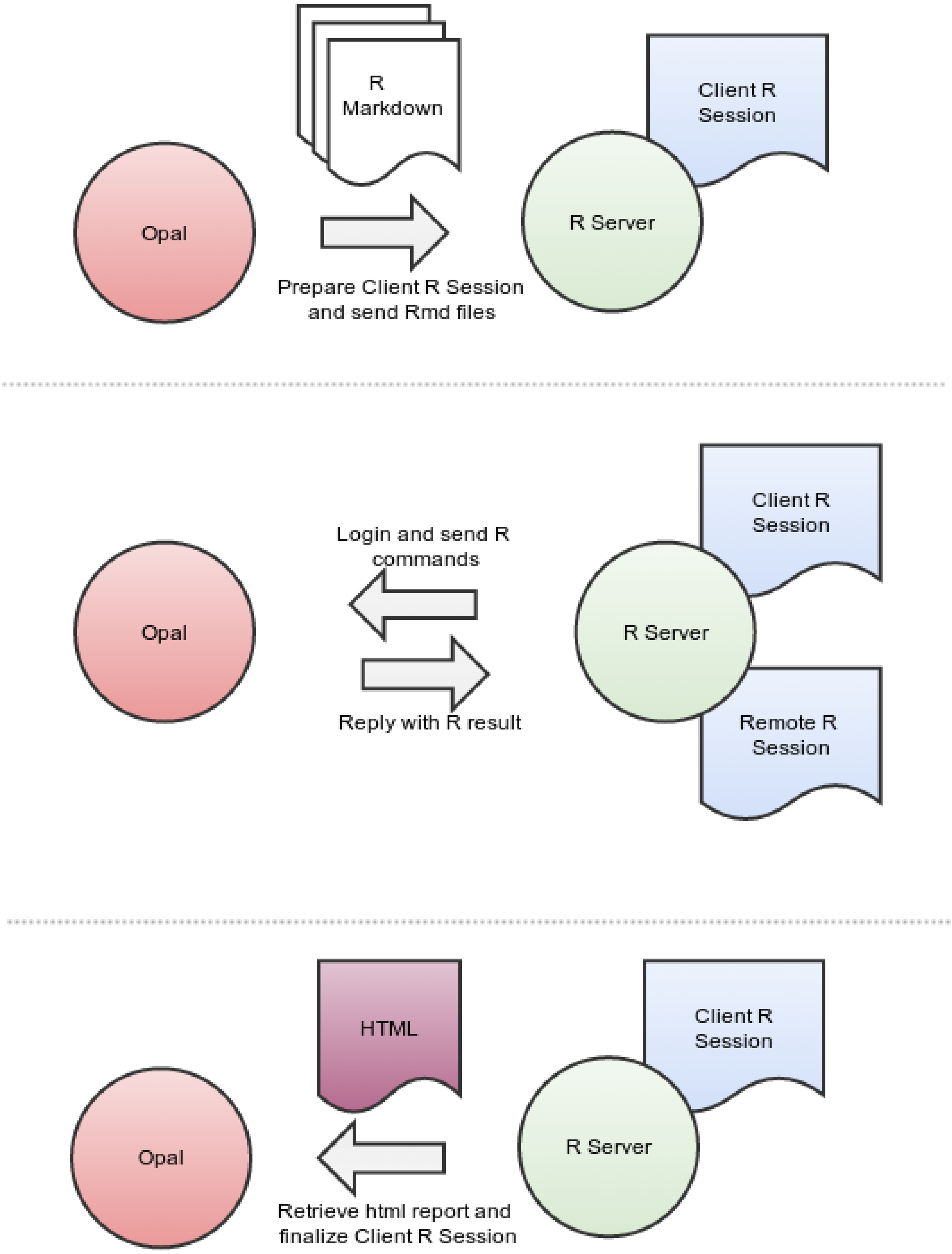
15.4.1 Local R Session

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



15.4.2 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

SIXTEEN

PYTHON INTRODUCTION

Opal Python client, a command line scripting tool (see *Python Commands*) and an API (see *Python API*) written in Python, enables automation of tasks in a Opal server.

16.1 Requirements

Python 3.7+ must be installed on the system. See more about Python.

16.2 Installation

The Opal Python Client is available on the official Python Package Index.

```
sudo pip install obiba-opal
```

Note: Previous versions were available as system packages. Make sure to remove them before installing the package with `pip`.

```
# on Debian systems
sudo apt-get remove opal-python-client

# on RPM systems
sudo yum remove opal-python-client
```

16.3 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

SEVENTEEN

PYTHON API

Opal Python can be easily extended by using the exposed classes. The classes *Command return an Opal task object, to be followed with the TaskService. The classes *Service perform immediate operations.

17.1 Packages

Use the pydoc tool to get the Python documentation.

```
pydoc obiba_opal.<package name>
```

Package	Description
obiba_opal.analysis	Table analysis related classes, to launch an analysis command and export analysis output.
obiba_opal.core	Core classes for establishing a connection with an Opal server, send requests and aget responses.
obiba_opal.data	Table data and entities related classes, to get list of entities, values and value sets.
obiba_opal.dictionary	Table dictionary related classes, to list tables and variables and to manage variable annotations.
obiba_opal.exports	Various export commands.
obiba_opal.file	File management classes, to upload, download, get info and delete files.
obiba_opal.imports	Various import commands.
obiba_opal.io	Import/export base classes.
obiba_opal.perm	Permission management related classes, for various Opal items.
obiba_opal.project	Project management classes, to perform backup/restore commands and to add/delete/list projects.
obiba_opal.security	Encryption/decryption service.
obiba_opal.sql	Table SQL related classes, to perform SQL queries on tables.
obiba_opal.subjects	Users and Groups management classes.
obiba_opal.system	Various system services, to perform operations on tasks, taxonomys, plugins etc.
obiba_opal.table	Backup/restore views.

17.2 Usage Example

```
from obiba_opal import OpalClient, HTTPError, Formatter, ImportCSVCommand, TaskService,␣
↪FileService, DictionaryService

# if 2-factor auth is enabled, user will be asked for the secret code
# Personal access token authentication is also supported (and recommended)
client = OpalClient.buildWithAuthentication(server='https://opal-demo.obiba.org', user=
↪'administrator', password='password')

try:
    # upload a local CSV data file into Opal file system
    fs = FileService(client)
    fs.upload_file('./data.csv', '/tmp')

    # import this CSV file into a project
    task = ImportCSVCommand(client).import_data('/tmp/data.csv', 'CNSIM')
    status = TaskService(client).wait_task(task['id'])

    # clean data file from Opal
    fs.delete_file('/tmp/data.csv')

    if status == 'SUCCEEDED':
        dico = DictionaryService(client)
        table = dico.get_table('CNSIM', 'data')
        # do something ...
        dico.delete_tables('CNSIM', ['data'])
    else:
        print('Import failed!')
        # do something ...
except HTTPError as e:
    Formatter.print_json(e.error, True)
finally:
    client.close()
```

CHAPTER EIGHTEEN

PYTHON COMMANDS

18.1 Project Commands

Project management to create/delete projects and backup/restore data.

18.1.1 Project

Manage a project.

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

Options

Option	Description
`--name NAME, -n NAME`	Project name. If no name and no action is specified, the list of the projects will be returned.
`--delete, -de`	Delete the project specified by the –name option
`--force, -f`	Skip confirmation on project deletion
`--add, -a`	Add a project specified by the –name option
`--database DATABASE, -db DATABASE`	Name of the database to be associated to the project at creation time (optional, unless you intend to import data).
`--title TITLE, -t TITLE`	The project's title at creation time (optional).
`--description DESCRIPTION, -dc DESCRIPTION`	The project's description at creation time (optional).
`--tags TAGS [TAGS ...], -tg TAGS [TAGS ...]`	The project tags at creation time (optional, separated by space).
`--export-folder EXPORT_FOLDER`	The project's preferred export folder (optional).

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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 projects with pretty JSON formatted output:

```
opal project --opal https://opal-demo.obiba.org --user administrator --password password␣
↪--json
```

Get a specific project:

```
opal project --opal https://opal-demo.obiba.org --user administrator --password password␣
↪--name CNSIM
```

Create the project foo without associated database:

```
opal project --opal https://opal-demo.obiba.org --user administrator --password password␣
↪--add --name foo
```

Delete the project foo:

```
opal project --opal https://opal-demo.obiba.org --user administrator --password password␣
↪--delete --name foo
```

18.1.2 Backup Project

The project backup task has a limited scope: tables (dictionary and data export), views (either as a logical table or as an exported table), resources, files and report templates. Other project elements that are not part of the backup: user and group permissions, view change history, table analysis, report executions etc.

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

Options

Option	Description
`--project PROJECT, -pr PROJECT`	Source project name
`--archive ARCHIVE, -ar ARCHIVE`	Archive directory path in the Opal file system
`--views-as-tables, -vt`	Treat views as tables, i.e. export data instead of keeping derivation scripts (default is false)
`--force, -f`	Force overwriting an existing backup folder

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

Example

Backup a specific project from a project, wait for the task to complete and download the archive as an encrypted zip file:

```
opal backup-project --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --project CNSIM --archive /home/administrator/backup/CNSIM | opal task --opal␣
↪https://opal-demo.obiba.org --user administrator --password password --wait && opal␣
↪file --download --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --download-password foobar123 /home/administrator/backup/CNSIM > CNSIM.zip
```

18.1.3 Restore Project

Restore the data of a project from a backup archive file to be found on the Opal file system. The destination project must exist and can have a name different from the original one (beware that this could break views). Default behavior is to stop when an item to restore already exist (override can be forced).

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

Options

Option	Description
`--project PROJECT, -pr PROJECT`	Destination project name
`--archive ARCHIVE, -ar ARCHIVE`	Archive directory or zip file path in the Opal file system
`--arpassword ARPASSWORD, -arp ARPASSWORD`	Password to decrypt zip archive (optional)
`--force, -f`	Force overwriting existing items (table, view, resource, report). Files override is not checked.

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

Example

Restore a project from the archive folder `CNSIM` into an existing project:

```
opal restore-project --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --project datashield --archive /home/administrator/backup/CNSIM
```

18.2 Datasources Commands

These commands allow to access to both variables and values.

18.2.1 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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

To fetch the description of a variable:

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

18.2.2 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

Example

Fetch the list of entity identifiers associated to a table:

```
opal data CNSIM.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 CNSIM.CNSIM1 -o https://opal-demo.obiba.org -u administrator -p password --id␣
↪1444 -j

# Get the JSON representation of a variable value
opal data CNSIM.CNSIM1:GENDER -o https://opal-demo.obiba.org -u administrator -p␣
↪password --id 1444 -j

# Get the raw value. If variable is of binary type, a byte stream is outputed.
opal data CNSIM.CNSIM1:GENDER -o https://opal-demo.obiba.org -u administrator -p␣
↪password --id 1444 -j --raw
```

18.2.3 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

Example

Fetch the entities with id 1444:

```
opal entity 1444 --opal https://opal-demo.obiba.org --user administrator --password password
```

Fetch the list of table where entity 1444 exists:

```
opal entity 1444 --opal https://opal-demo.obiba.org --user administrator --password password --tables
```

Fetch the list of table where entity 1444 of type “Participant” exists:

```
opal entity 1444 --opal https://opal-demo.obiba.org --user administrator --password password --tables --type Participant
```

18.2.4 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user’s personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.2.5 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	Credentials auth: user name (requires a password)
--password PASSWORD, -p PASSWORD	Credentials auth: user password (requires a user name)
--token TOKEN, -tk TOKEN	Token auth: user's personal access token
--ssl-cert SSL_CERT, -sc SSL_CERT	Two-way SSL auth: certificate/public key file (requires a private key)
--ssl-key SSL_KEY, -sk SSL_KEY	Two-way SSL auth: private key file (requires a certificate)

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
```

18.2.6 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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
```

18.2.7 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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
```

18.2.8 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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
```

18.2.9 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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
```

18.2.10 SQL

Execute a SQL query on one or more tables of a project. Permission to access values of these tables is required.

See more SQL query examples in the Project Tables *SQL* section.

```
opal sql <CREDENTIALS> [OPTIONS] [XTRAS]
```

Options

Option	Description
`--project PROJECT, -pr PROJECT`	Source project name, that will be used to resolve the table names in the FROM statement. If not provided, the fully qualified table names must be specified in the query (escaped by backquotes: \`<project>.<table>\`).
`--query QUERY, -q QUERY`	The SQL query
`--format FORMAT, -f FORMAT`	The format of the output, can be `json` or `csv`. Default is `csv`.
`--id-name ID_NAME, -in ID_NAME`	Name of the ID column name. Default is `_id`.

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

Example

Simple SQL query with CSV output:

```
opal sql --opal https://opal-demo.obiba.org --user administrator --password password --
↪project CNSIM --query "select * from CNSIM1 limit 10"
```

Simple SQL query with JSON output:

```
opal sql --opal https://opal-demo.obiba.org --user administrator --password password --
↪project CNSIM --query "select * from CNSIM1 limit 10" --format json
```

More advanced SQL query:

```
opal sql --opal https://opal-demo.obiba.org --user administrator --password password --
↪project CNSIM --query "select count(*) as N, avg(LAB_HDL) as HDL_AVG, GENDER from␣
↪(select * from CNSIM1 union all select * from CNSIM2) where LAB_HDL is not null group␣
↪by GENDER"
```

Simple SQL query with CSV output and without specifying a project in the arguments:

```
opal sql --opal https://opal-demo.obiba.org --user administrator --password password --
↪query "select * from `CNSIM.CNSIM1` limit 10"
```

You can also escape the backticks (`) when executing from a Linux terminal (Bash)

```
opal sql --opal https://opal-demo.obiba.org --user administrator --password password --
↪query "select * from \`CNSIM.CNSIM1\` limit 10"
```

18.2.11 SQL History

Extract the list of the SQL queries executed, with their error status (if any), start/end time and project context (if any). Regular users can only retrieve their own queries, whereas administrators can retrieve queries from all or any users.

```
opal sql-history <CREDENTIALS> [OPTIONS] [XTRAS]
```

Options

Option	Description
`--project PROJECT, -pr PROJECT`	Project name used as the SQL execution context to filter. If not specified, history from any context is returned. If '*' is specified, history of SQL execution without context is returned.
`--offset OFFSET, -os OFFSET`	Number of history items to skip. Default is 0 (note that the items are ordered by most recent first).
`--limit LIMIT, -lm LIMIT`	Maximum number of history items to return. Default is 100.
`--subject SUBJECT, -sb SUBJECT`	Filter by user name, only administrators can retrieve SQL history of other users. If '*' is specified, history of all users is retrieved. Default is the current user name.

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user’s personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

Example

Get own SQL queries history:

```
opal sql-history --opal https://opal-demo.obiba.org --user administrator --password␣
↪password
```

Get own SQL queries history on project CNSIM:

```
opal sql-history --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --project CNSIM
```

Get other user SQL queries history on project CNSIM (requires administrator rights):

```
opal sql-history --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --subject someuser --project CNSIM
```

Get all users, one thousand most recent SQL queries history (requires administrator rights):

```
opal sql-history --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --subject '*' --limit 1000
```

18.3 Import Commands

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

18.3.1 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.
`--merge, -mg`	Merge imported data dictionary with the destination one (default is false, i.e. data dictionary is overridden).

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.3.2 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
`--merge, -mg`	Merge imported data dictionary with the destination one (default is false, i.e. data dictionary is overridden).

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.3.3 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.
`--merge, -mg`	Merge imported data dictionary with the destination one (default is false, i.e. data dictionary is overridden).

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.3.4 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.
`--merge, -mg`	Merge imported data dictionary with the destination one (default is false, i.e. data dictionary is overridden).

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.3.5 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.
`--merge, -mg`	Merge imported data dictionary with the destination one (default is false, i.e. data dictionary is overridden).

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.3.6 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.
`--merge, -mg`	Merge imported data dictionary with the destination one (default is false, i.e. data dictionary is overridden).

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user’s personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.3.7 Import RDS (R)

Import a RDS file, to be found in Opal file system, using R. A RDS file contains a single serialized R object, which is expected to be a tibble and that can be produced in R using base::saveRDS().

```
opal import-r-rds <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 RDS file to import on the Opal file system
`--type TYPE, -ty TYPE`	Entity type (default is Participant)
`--idVariable IDVARIABLE, -iv IDVARIABLE`	tibble's column name that provides the entity ID. If not specified, first column values are considered to be the entity identifiers.
`--merge, -mg`	Merge imported data dictionary with the destination one (default is false, i.e. data dictionary is overridden).

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

Example

Import table from CNSIM1 RDS file in CNSIM project:

```
opal import-r-rds --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --destination CNSIM --path /home/administrator/CNSIM1.rds
```

18.3.8 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 (exclusive from using token)
`--rpassword RPASSWORD, -rp RPASSWORD`	Remote user password (exclusive from using token)
`--rtoken RTOKEN, -rt RTOKEN`	Remote personal access token (exclusive from user credentials)
`--rdatasource RDATASOURCE, -rd RDATASOURCE`	Remote datasource name
`--merge, -mg`	Merge imported data dictionary with the destination one (default is false, i.e. data dictionary is overridden).

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.3.9 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)
`--merge, -mg`	Merge imported data dictionary with the destination one (default is false, i.e. data dictionary is overridden).

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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)"
```

18.3.10 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
`--merge, -mg`	Merge imported data dictionary with the destination one (default is false, i.e. data dictionary is overridden).

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.3.11 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user’s personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.3.12 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user’s personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.3.13 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.4 Export Commands

Export one or more tables to the Opal file system.

18.4.1 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
`--id-name ID_NAME, -in ID_NAME`	ID column name (optional)
`--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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.4.2 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	Credentials auth: user name (requires a password)
--password PASSWORD, -p PASSWORD	Credentials auth: user password (requires a user name)
--token TOKEN, -tk TOKEN	Token auth: user’s personal access token
--ssl-cert SSL_CERT, -sc SSL_CERT	Two-way SSL auth: certificate/public key file (requires a private key)
--ssl-key SSL_KEY, -sk SSL_KEY	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.4.3 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
`--id-name ID_NAME, -in ID_NAME`	ID column name (optional)
`--output OUTPUT, -out OUTPUT`	Output SAS or SAS Transport file name on the Opal file system

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.4.4 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
`--id-name ID_NAME, -in ID_NAME`	ID column name (optional)
`--output OUTPUT, -out OUTPUT`	Output SPSS or compressed SPSS file name on the Opal file system

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.4.5 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
`--id-name ID_NAME, -in ID_NAME`	ID column name (optional)
`--output OUTPUT, -out OUTPUT`	Output Stata file name on the Opal file system

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.4.6 Export RDS (R)

Export in RDS format in Opal file system. A RDS file contains a single serialized R object, which will be a tibble and that can be read in R using base::readRDS().

```
opal export-r-rds <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
`--id-name ID_NAME, -in ID_NAME`	ID column name (optional)
`--output OUTPUT, -out OUTPUT`	Output RDS file name on the Opal file system

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

Example

Export table from opal-data to a RDS file:

```
opal export-r-rds --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --datasource CNSIM --tables CNSIM1 --output /tmp/cnsim1.rds
```

18.4.7 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.4.8 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	Credentials auth: user name (requires a password)
--password PASSWORD, -p PASSWORD	Credentials auth: user password (requires a user name)
--token TOKEN, -tk TOKEN	Token auth: user's personal access token
--ssl-cert SSL_CERT, -sc SSL_CERT	Two-way SSL auth: certificate/public key file (requires a private key)
--ssl-key SSL_KEY, -sk SSL_KEY	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.5 User and Group Commands

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

18.5.1 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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
```

18.5.2 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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
```

18.6 Permission Commands

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

18.6.1 Project Permission

Manage global permissions on the project.

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

Options

Option	Description
`--fetch, -f`	Fetch permissions
`--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 (required on add/delete)
`--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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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
```

18.6.2 Datasource Permission

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

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

Options

Option	Description
`--fetch, -f`	Fetch permissions
`--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 (required on add/delete)
`--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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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
```

18.6.3 Table Permission

Manage permissions on a project's table.

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

Options

Option	Description
`--fetch, -f`	Fetch permissions
`--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 (required on add/delete)
`--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 get/set (default is all)

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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 --tables 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
```

18.6.4 Resources Permission

Manage permissions on a project's resources as a whole. These permissions apply to any resources in the project without needing to name them: after adding a resource the same permission will apply.

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

Options

Option	Description
`--fetch, -f`	Fetch permissions
`--add, -a`	Add a permission
`--delete, -d`	Delete a permission
`--permission PERM, -pe PERM`	Permission to apply: `view`, `administrate`
`--subject SUBJECT, -s SUBJECT`	Subject name to which the permission will be granted (required on add/delete)
`--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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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 any resource in RSRC project:

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

Remove the above permission:

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

18.6.5 Resource Permission

Manage permissions on a project's resource.

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

Options

Option	Description
`--fetch, -f`	Fetch permissions
`--add, -a`	Add a permission
`--delete, -d`	Delete a permission
`--permission PERM, -pe PERM`	Permission to apply: `view`, `administrate`
`--subject SUBJECT, -s SUBJECT`	Subject name to which the permission will be granted (required on add/delete)
`--type TYPE, -ty TYPE`	Subject type: `user` or `group`
`--project PROJECT, -pr PROJECT`	Project name on which the permission is to be set
`--resources RESOURCE [RESOURCE ...], -r RESOURCE [RESOURCE ...]`	List of resource names on which the permission is to be get/set (default is all)

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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 resource CNSIM1 in RSRC project:

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

Remove the above permission:

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

Add permission on all resources of RSRC project:

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

Remove permission from all resource of RSRC project:

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

Add permission on specific resources:

```
opal perm-resource --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --type USER --project RSRC --subject demouser --permission view --add --
↪resources CNSIM1 CNSIM2
```

18.6.6 Variable Permission

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

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

Options

Option	Description
`--fetch, -f`	Fetch permissions
`--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 (required on add/delete)
`--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 get/set (default is all)

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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
```

18.6.7 System Permission

Manage global system permissions.

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

Options

Option	Description
`--fetch, -f`	Fetch permissions
`--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 (required on add/delete)
`--type TYPE, -ty TYPE`	Subject type: `user` or `group`

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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
```

18.6.8 R Permission

Manage R server usage permissions.

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

Options

Option	Description
`--fetch, -f`	Fetch permissions
`--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 (required on add/delete)
`--type TYPE, -ty TYPE`	Subject type: `user` or `group`

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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
```

18.6.9 DataSHIELD Permission

Manage DataSHIELD usage permissions.

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

Options

Option	Description
`--fetch, -f`	Fetch permissions
`--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 (required on add/delete)
`--type TYPE, -ty TYPE`	Subject type: `user` or `group`

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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
```

18.7 Other Commands

Other commands, for advanced users.

18.7.1 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 (8 characters minimum).
--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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
--opal OPAL, -o OPAL	Opal server base url
--user USER, -u USER	Credentials auth: user name (requires a password)
--password PASSWORD, -p PASSWORD	Credentials auth: user password (requires a user name)
--token TOKEN, -tk TOKEN	Token auth: user's personal access token
--ssl-cert SSL_CERT, -sc SSL_CERT	Two-way SSL auth: certificate/public key file (requires a private key)
--ssl-key SSL_KEY, -sk SSL_KEY	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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 (encrypted zip file) from Opal file system:

```
opal file --opal https://opal-demo.obiba.org --user administrator --password password --
↪download-password foobar123 /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
```

18.7.2 Taxonomy

Manage taxonomies: list a summary of the available taxonomies if no option is provided, otherwise download, import or delete a taxonomy.

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

Options

Option	Description
`--download DOWNLOAD, -dl DOWNLOAD`	Download a taxonomy from the provided name, in YAML format
`--import-file IMPORT_FILE, -if IMPORT_FILE`	Import a taxonomy from a YAML file in Opal file system. Fail if the taxonomy already exists.
`--delete DELETE, -dt DELETE`	Delete a taxonomy from the provided name
`--force, -f`	Skip taxonomy deletion confirmation

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

Example

List the taxonomies:

```
opal taxonomy --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --json
```

Download a taxonomy in YAML format:

```
opal taxonomy --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --download MyTaxo > MyTaxo.yml
```

Upload a local taxonomy file (in YAML format) into the Opal file system and import it:

```
opal file --opal https://opal-demo.obiba.org --user administrator --password password --
↪upload /path/to/MyTaxo.yml /home/administrator
opal taxonomy --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --import-file /home/administrator/MyTaxo.yml
```

Delete a taxonomy:

```
opal taxonomy --opal https://opal-demo.obiba.org --user administrator --password␣
↪password --delete MyTaxo
```

18.7.3 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.7.4 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

Example

List the plugins that are currently installed:

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

18.7.5 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 can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

Extras

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

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
```

18.7.6 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: { "Key1": "Value1", "Key2": "Value2" }

Credentials

Authentication can be done by username/password credentials OR by personal access token OR by certificate/private key pair (two-way SSL authentication).

Option	Description
`--opal OPAL, -o OPAL`	Opal server base url
`--user USER, -u USER`	Credentials auth: user name (requires a password)
`--password PASSWORD, -p PASSWORD`	Credentials auth: user password (requires a user name)
`--token TOKEN, -tk TOKEN`	Token auth: user's personal access token
`--ssl-cert SSL_CERT, -sc SSL_CERT`	Two-way SSL auth: certificate/public key file (requires a private key)
`--ssl-key SSL_KEY, -sk SSL_KEY`	Two-way SSL auth: private key file (requires a certificate)

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 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

NINETEEN

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.

19.1 Syntax

19.1.1 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.

19.1.2 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()
```

19.2 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:

19.2.1 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 {
```

(continues on next page)

(continued from previous page)

```
   log('Collected tubes matches expected tubes.');
}
```

19.2.2 switch

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

19.3 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

TWENTY

MAGMA JS METHODS

20.1 Global Methods

20.1.1 $, $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.

20.1.2 $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')
```

20.1.3 $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	

20.1.4 $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
$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()
```

20.1.5 $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'))
```

20.1.6 $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')
```

20.1.7 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'));
```

20.1.8 now

Returns the current date time wrapped in a value object.

Syntax

```
now()
```

Examples

Get the current date time.

```
now()
```

20.1.9 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')

// 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())
```

20.1.10 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'])
```

20.1.11 $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'))
```

20.1.12 $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'))
```

20.1.13 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());
```

20.2 Variable Methods

20.2.1 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')
```

20.2.2 name

Get the name of the variable as a Text value.

See also *$var, $variable*.

Syntax

```
name()
```

Examples

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

20.2.3 type

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

See also *$var, $variable*.

Syntax

```
type()
```

Examples

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

20.2.4 entityType

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

See also *$var, $variable*.

Syntax

```
entityType()
```

Examples

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

20.2.5 refEntityType

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

See also *$var, $variable*.

Syntax

```
refEntityType()
```

Examples

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

20.2.6 repeatable

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

See also *$var, $variable*.

Syntax

```
repeatable()
```

Examples

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

20.2.7 occurrenceGroup

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

See also *$var, $variable*.

Syntax

```
occurrenceGroup()
```

Examples

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

20.2.8 mimeType

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

See also *$var, $variable*.

Syntax

```
mimeType()
```

Examples

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

20.2.9 unit

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

See also *$var, $variable*.

Syntax

```
unit()
```

Examples

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

20.2.10 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()
```

20.2.11 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()
```

20.2.12 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()
```

20.2.13 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()
```

20.3 Value Methods

20.3.1 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')
```

20.3.2 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()
```

20.3.3 isNull

Returns true if the value is null.

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

Syntax

```
isNull()
```

Examples

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

20.3.4 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.
$('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');
```

(continues on next page)

(continued from previous page)

```
// Applies to each value in a value sequence
$('MyRepeatableVar').whenNull(99);
```

20.3.5 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) })
```

20.3.6 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'))
```

20.3.7 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

20.3.8 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()
// 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]
```

20.3.9 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()
```

20.4 Value Sequence Methods

20.4.1 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()
```

20.4.2 first

Returns the first value in a value sequence.

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

Syntax

```
first()
```

Examples

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

20.4.3 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()
```

20.4.4 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')
```

20.4.5 last

Returns the last value in a value sequence.

See also *valueAt*, *first*.

Syntax

```
last()
```

Examples

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

20.4.6 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')
```

20.4.7 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')
```

20.4.8 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()
```

20.4.9 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)
```

20.4.10 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();
})
```

20.4.11 unique

Filter the unique values of a value sequence.

See also *filter*.

Syntax

```
unique()
```

Examples

```
$('LAB_HDL').unique()
```

20.4.12 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;
})
```

20.4.13 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();
})
```

20.4.14 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();
})
```

20.4.15 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();
```

20.4.16 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();
```

20.4.17 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();
```

20.4.18 median

Returns the median value from a value sequence. The returned value will have the decimal value type.

If the sequence is empty or null, null is returned. All the null values of the sequence are excluded from the calculation. This method also accepts a non-sequence value, in which case, it is returned untouched.

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

Syntax

```
median()
```

Examples

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

20.4.19 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();
```

20.4.20 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();
```

20.4.21 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();
```

20.4.22 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);
```

20.4.23 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);
```

20.4.24 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').instertAt(0, $('StandingHeight:FIRST_RES_PULSE')).
↪avg();

// Insert several values to a value to produce the sequence: a, b, c, null
newValue('a').insertAt(1, 'b', 'c', null);

// Insert several values to a value to produce the sequence: a, null, b, c, null
newValue('a').insertAt(2, 'b', 'c', null);

// Insert several values to a value sequence to produce the sequence: a, c, d, null, b
newSequence(['a', 'b']).insertAt(1, 'c', 'd', null);
```

20.4.25 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();
```

20.4.26 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);
  });
```

20.4.27 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()
```

20.4.28 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()
```

20.5 Boolean Value Methods

20.5.1 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'))
```

20.5.2 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'));
```

20.5.3 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');
```

20.5.4 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'))
```

20.6 Numeric Value Methods

20.6.1 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)
```

20.6.2 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)
```

20.6.3 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)
```

20.6.4 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)
```

20.6.5 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)
```

20.6.6 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)
```

20.6.7 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)
```

20.6.8 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)
```

20.6.9 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()
```

20.6.10 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
```

20.6.11 abs

Returns the absolute value of the current value.

Syntax

```
abs()
```

Examples

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

20.6.12 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
```

20.6.13 sqroot

Returns the square root of the value.

See also *root*, *cbroot*.

Syntax

```
sqroot()
```

Examples

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

20.6.14 cbroot

Returns the cubic root of the value.

See also *root*, *sqroot*.

Syntax

```
cbroot()
```

Examples

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

20.6.15 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()
```

20.6.16 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)
```

20.6.17 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
 });
```

20.7 Text Value Methods

20.7.1 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'));
```

20.7.2 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'));
```

20.7.3 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")
```

20.7.4 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")
```

20.7.5 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/)
```

20.7.6 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')
```

20.7.7 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();
```

20.7.8 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')
```

20.7.9 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')
```

20.7.10 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(':;,( .["')
```

20.8 Date and Datetime Value Methods

20.8.1 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)
```

20.8.2 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'))
```

20.8.3 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'))
```

20.8.4 dayOfMonth

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

See also *dayOfWeek*, *dayOfYear*.

Syntax

```
dayOfMonth()
```

Examples

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

20.8.5 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()
```

20.8.6 dayOfYear

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

See also *dayOfMonth*, *dayOfWeek*.

Syntax

```
dayOfYear()
```

Examples

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

20.8.7 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'))
```

20.8.8 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()
```

20.8.9 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()
```

20.8.10 minute

Returns the minute within the hour.

Syntax

```
minute()
```

Examples

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

20.8.11 millisecond

Returns the millisecond within the second.

Syntax

```
millisecond()
```

Examples

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

20.8.12 month

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

Syntax

```
month()
```

Examples

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

20.8.13 quarter

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

Syntax

```
quarter()
```

Examples

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

20.8.14 second

Returns the second within the minute.

Syntax

```
second()
```

Examples

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

20.8.15 semester

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

Syntax

```
semester()
```

Examples

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

20.8.16 time

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

Syntax

```
time()
```

Examples

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

20.8.17 weekday

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

Syntax

```
weekday()
```

Examples

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

20.8.18 weekend

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

Syntax

```
weekend()
```

Examples

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

20.8.19 weekOfMonth

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

Syntax

```
weekOfMonth()
```

Examples

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

20.8.20 weekOfYear

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

Syntax

```
weekOfYear()
```

Examples

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

20.8.21 year

Returns the year value.

Syntax

```
year()
```

Examples

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

20.9 Geo Value Methods

20.9.1 latitude

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

See also *longitude*.

Syntax

```
latitude()
```

Examples

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

20.9.2 longitude

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

See also *latitude*.

Syntax

```
longitude()
```

Examples

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

20.10 Measurement Unit Methods

20.10.1 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'
```

20.10.2 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.
unique	Returns a sequence of distinct values.
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.

Table 1 – continued from previous page

Methods	Description
min	Returns the minimum value of a value sequence.
avg	Returns the average of a value sequence.
median	Returns the median value from 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
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.

CHAPTER

TWENTYONE

PARTNERS AND FUNDERS

The development of this application was made possible thanks to the support of our partners and funders:

mælstrom	Epigeny	canarie	EUCAN connect

CHAPTER TWENTYTWO

SUPPORT

Please visit OBiBa support page.