Mica Documentation

OBiBa

Jan 17, 2024

CONTENTS

1 Introduction **3**
- 1.1 Mica Server . . . 3
- 1.2 Opal Server . . . 5
- 1.3 Agate Server . . . 5

2 Documents **7**
- 2.1 Types . . . 7
- 2.2 Search . . . 9
- 2.3 Associations . . . 9
- 2.4 Annotations . . . 10
- 2.5 Permissions . . . 11
- 2.6 Revision History . . . 11
- 2.7 Comments . . . 11
- 2.8 Cart . . . 11

3 Publication Flow **13**
- 3.1 Revision Status . . . 13
- 3.2 Transitions . . . 15

4 Installation **17**
- 4.1 Requirements . . . 17
- 4.2 Install . . . 17
- 4.3 Upgrade . . . 20
- 4.4 Execution . . . 20

5 Configuration **23**
- 5.1 HTTP Server Configuration . . . 23
- 5.2 MongoDB Server Configuration . . . 23
- 5.3 Opal Server Configuration . . . 24
- 5.4 Agate Server Configuration . . . 25
- 5.5 Cross Site Resource Forgery (CSRF) . . . 25
- 5.6 Shiro Configuration . . . 25
- 5.7 Elasticsearch Configuration . . . 25
- 5.8 Miscelaneous Configuration . . . 27
- 5.9 User Directories . . . 27
- 5.10 Reverse Proxy Configuration . . . 28

6 Public Pages Configuration **31**
- 6.1 Page Templates . . . 31
- 6.2 Theme and Style . . . 39
- 6.3 Translations . . . 41

7 Plugins **43**
7.1 Repository . . . 43
7.2 Installation . . . 45
7.3 Configuration . . . 45
7.4 Backups . . . 46

8 Backup and Restore **47**
8.1 Mica . . . 47
8.2 MongoDB . . . 48
8.3 Using Docker . . . 48

9 Web Introduction **49**
9.1 Requirements . . . 49

10 Documents Management **51**
10.1 View, Edit, Publish . . . 51
10.2 Revisions . . . 53
10.3 Files . . . 54
10.4 Comments . . . 56
10.5 Permissions . . . 57

11 Data Access Requests Management **59**
11.1 Summary . . . 59
11.2 Request Workflow . . . 59
11.3 Forms . . . 63
11.4 Documents . . . 64
11.5 Comments . . . 64
11.6 Private Comments . . . 64
11.7 History . . . 64
11.8 Collaborators . . . 64
11.9 Notification Emails . . . 64

12 Administration **67**
12.1 System . . . 67
12.2 Content . . . 75
12.3 Data Discovery . . . 81

13 Python Introduction **85**
13.1 Requirements . . . 85
13.2 Installation . . . 85
13.3 Usage . . . 86

14 Authorization Commands **87**
14.1 Document Access . . . 87
14.2 File Access . . . 89
14.3 Document Permission . . . 90

15 Document Commands **93**
15.1 Update Collected Dataset . . . 93
15.2 Update Collected Datasets . . . 94
15.3 File Management . . . 96
15.4 Search . . . 97
15.5 Import Zip . . . 99

16 Other Commands **101**

16.1 Plugin . . . 101
16.2 Web Services . . . 102

17 R Introduction **105**
17.1 Requirements . . . 105
17.2 Installation . . . 105

18 R Usage **107**
18.1 Connection . . . 107
18.2 Search . . . 107

19 REST API Introduction **109**
19.1 Authentication . . . 109
19.2 Authorization . . . 109
19.3 Clients . . . 109

20 Draft Documents **111**
20.1 Draft Networks . . . 111
20.2 Draft Network . . . 114

21 Partners and Funders **123**

22 Support **125**

HTTP Routing Table **127**

Targeted at individual studies and study consortia, OBiBa software stack (Opal, Mica etc.) provides a software solution for epidemiological data management, analysis and publication. While Opal, the core data warehouse application, provides all the necessary tools to import, transform and describe data, Mica provides everything needed to build personalized web data portals and publish content of research activities of both studies and consortia. Mica is to be used with Agate, the OBiBa's central authentication server which centralizes user related services such as profile management, and a notification system using emails.

CHAPTER

ONE

INTRODUCTION

Mica is an advanced web application designed to create data web portals for large-scale epidemiological studies or multiple-study consortia. It provides a structured description of consortia, studies, annotated and searchable data dictionaries, and data access request management.

Mica is built upon a multi-tier architecture consisting of several RESTful server and client applications. The table below list each application with a brief description:

Application	Description
Mica Server	Java server providing web services (REST) for managing, storing, searching Mica Domain content and communicating with other servers listed below.
Opal Server	Optional Java server providing web services (REST) for importing, transforming and analyzing study variables.
Agate Server	Java server providing web services (REST) for user management and notifications.
Mica Web Application	Front-end to Mica Server providing client interface to manage Mica Domain content as well as to administrate and configure access permissions and secure connections.
Mica Python Client	Python front-end to Mica server providing services for administrative command-line and automation tasks.
Mica R Client	R front-end to Mica server providing services for Mica content analysis and reporting.

The diagram below illustrates the relationships between the Mica server and the other tiers:

The system architecture can be simplified by replacing Opal with data dictionary and taxonomy files (or remote services using *Plugins*):

Note that it is possible to combine Opal with data dictionary and taxonomy files.

1.1 Mica Server

Editors and reviewers of the Mica web portal content can access to the web interface of this server as described in the Mica Web Application User Guide. Data access request form can also be configured through this web interface.

Mica server is a client of Opal and Agate servers.

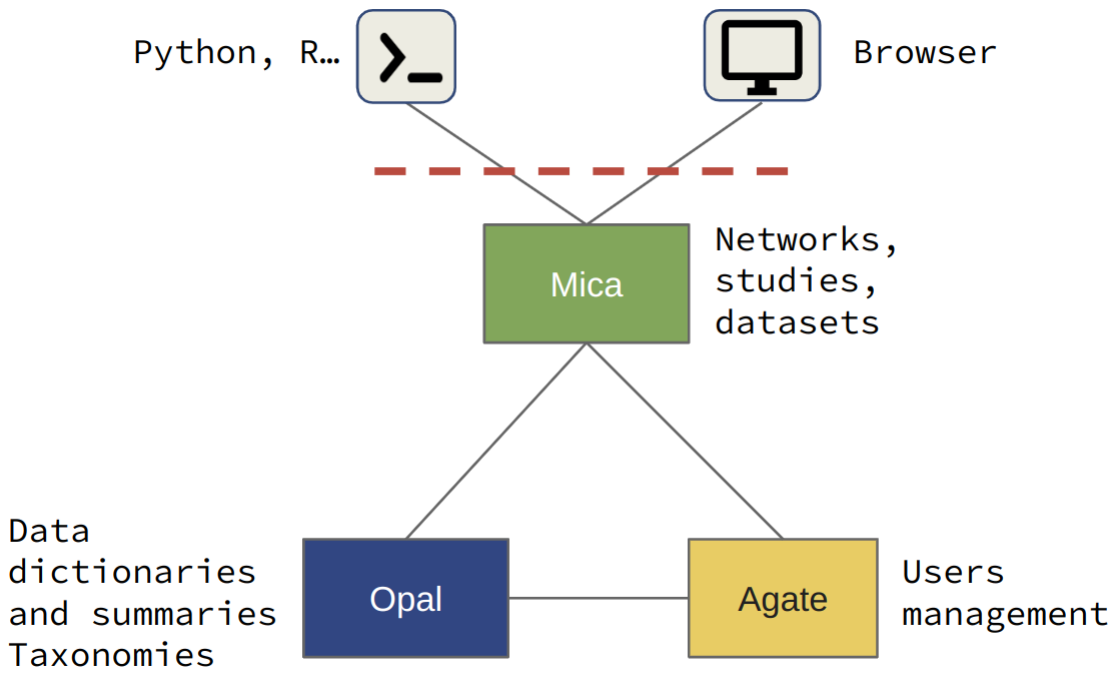


Fig. 1: Mica system regular architecture

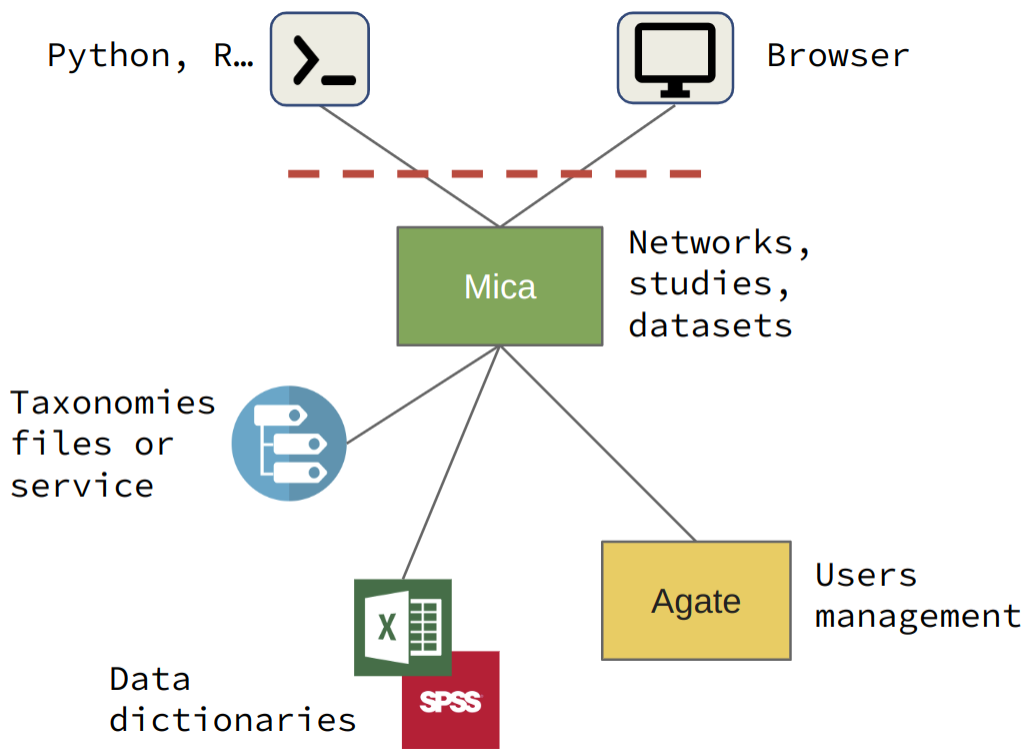


Fig. 2: Mica system light architecture

1.2 Opal Server

Note: Since Mica 5.2 it is not required to manage data dictionaries and taxonomies in a Opal server. The data dictionaries can be provided by Excel files attached to the datasets. The taxonomies can be defined by local configuration files or can be downloaded from a remote service (using an appropriate plugin).

Opal application is used for:

- defining data dictionaries (variables),
- storing data,
- providing data summary statistics.

Opal offers well established security controls, allowing to NOT expose individual-level data. Note also that the Opal server is only accessed by the Mica server, reducing the risk of data compromisation from a malicious end user.

Installation and configuration guides can be found in the Opal documentation.

Mica expects at least one Opal server when some datasets are defined. Additional Opal servers can also be identified to access to distributed datasets.

1.3 Agate Server

Agate application is used for:

- having a user directory shared between OBiBa's applications,
- having centralized services such as profile management and email notifications.

Installation and configuration guides can be found in the Agate documentation.

CHAPTER

TWO

DOCUMENTS

Mica handles several type of documents, specific to the epidemiological studies domain: network, study, datasets etc. These document types have their own internal structure (to allow relationships between them and to ensure basic search), but can also be extended with custom fields. The default set of fields is the one promoted by Maelstrom Research. This default description model should fit with your needs in most of the cases.

All the documents follow the *Publication Flow* except the *Data Access Request* (which is a form privately exchanged between a researcher and the study/consortium).

2.1 Types

2.1.1 Network

A network is a group of epidemiological studies that has specific research interests. It is described using the following fields: name, aims, investigators, contact information and participating studies. It can also be related to other networks.

2.1.2 Individual Study

An individual study is defined as any epidemiological study (e.g. cohort, case control, cross sectional, etc.) conducted to better understand the distribution and determinants of health and disease. It is described using the following fields: name, objectives, investigators, contact information, design, data collection timeline, target number and characteristics of participants, and related scientific publications and documents. A study can include one or more populations described below.

Population

A population is a set of individuals sharing the same selection criteria for enrollment in a study. It is described using the following fields: name, sources of recruitment, participant characteristics, and number of participants. A population is linked to one or more data collection events according to the number of follow-ups.

Data Collection Event

A data collection event is a collection of information on one or more population(s) over a specific period of time (e.g. baseline, follow-up 1, follow-up 2). It is described using the following fields: name, start and end date, and data sources (e.g. questionnaires, physical measures, biosample measures, etc). A data collection event may be associated to one or more populations and it can include one or more datasets.

2.1.3 Harmonization Initiative

A harmonization initiative is defined as a research project harmonizing data across individual studies to answer specific reseach questions. It is described using the following fields: acronym, contact information, objectives, start year, end year and related documents. A harmonization initiative can include one or more harmonization protocols (dataschema).

2.1.4 Collected Dataset

A collected (study-specific) dataset holds metadata about the variables collected within a data collection event. The metadata is described using a standardized format of data dictionary which provides information on collected variables' definitions and characteristics (e.g. type, unit, categories, and area of information covered). It can be associated to a study by specifying a data collection event.

Collected Variable

A collected variable is a variable that was collected, measured, or constructed within a study protocol. It is described using the following fields: name, label, description, type, unit, categories, and area of information covered. If the collected dataset includes data, summary statistics of the collected variable can be published on the web portal (e.g. means, minimum, maximum, counts and percentages). Each collected variable is part of one and only one study-specific dataset.

2.1.5 Harmonization Protocol

A harmonization protocol holds metadata about core variables constructed from multiple collected datasets. The metadata is described using a standardized format of data dictionary which provides information on harmonized variables' definitions and characteristics (e.g. type, unit, categories, and area of information covered): this represent the data schema of the harmonization protocol. It can be optionally associated to the harmonized data.

Data Schema Variable

A data schema variable is the harmonization protocol reference variable. Each harmonized variable will *implement* a corresponding data schema variable.

Harmonized Variable

A harmonized variable is a core variable (common format) generated by multiple individual studies. It is described using the following fields: name, label, description, type, unit, categories, and area of information covered. If the harmonization protocol includes data, summary statistics of the harmonized variable (e.g. means, minimum, maximum, counts and percentages) can be published on the web portal. Each harmonized variable is part of one and only one harmonization protocol.

2.1.6 Research Project

A research project reports information about the work that was conducted thanks to the network/study data: research objectives and results, contact information, status timeline. It could be somehow related to a data access request but not necessarily.

2.1.7 Data Access Request

A data access request is an application form that researchers submit in order to gain access to the network/study data. There is a predefined workflow from submission to approval described in more detail in *Data Access Requests Management*.

2.1.8 Data Access Amendment

A data access amendment is an application form that researchers submit to request changes to a pre-approved data access request. The data access amendment workflow is identical to that of data access requests.

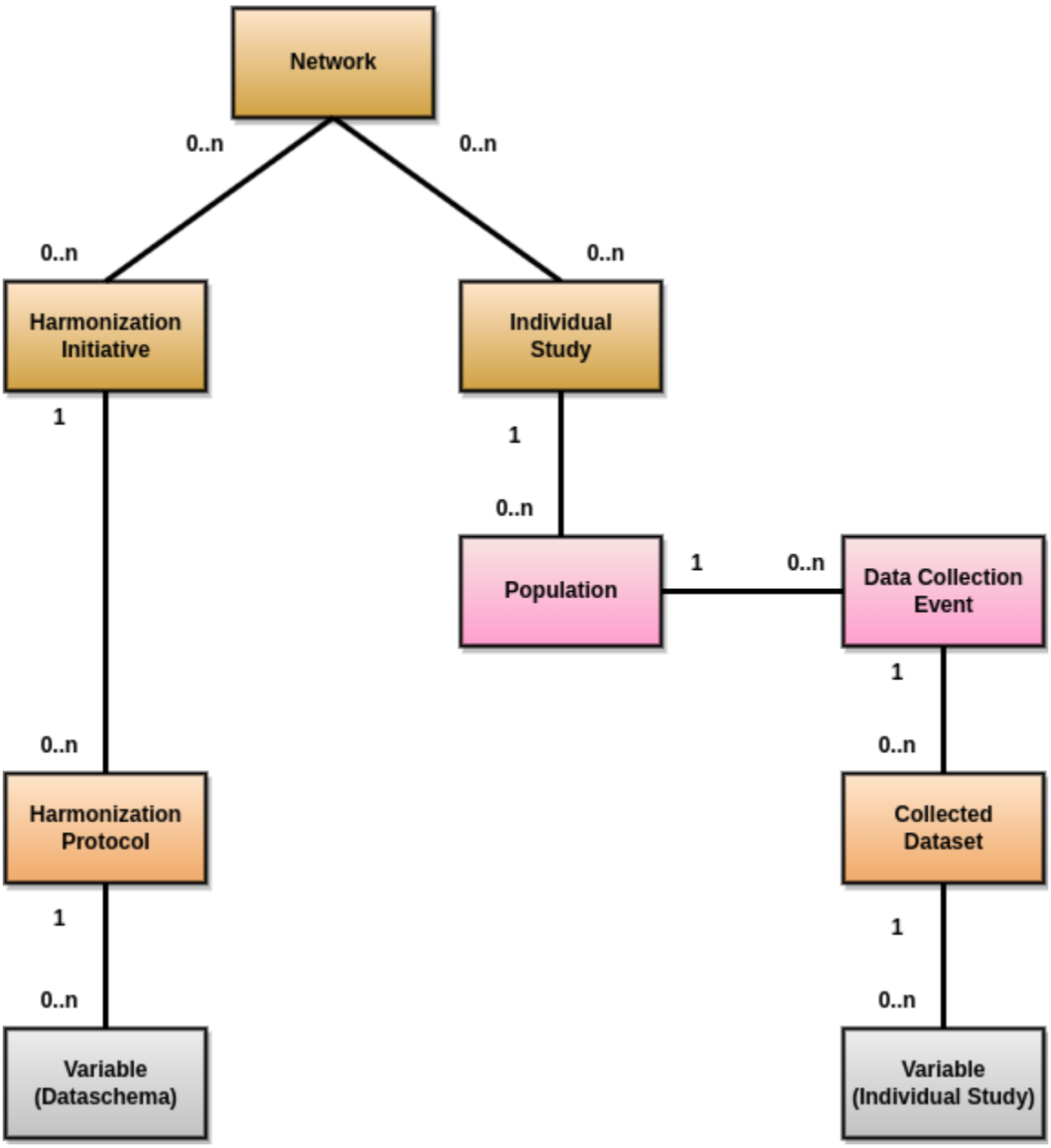
2.2 Search

Mica search engine allows to look into the domain while applying criteria on each type of document. The result of this combined query can be of any type. For example:

- search for variables about alcohol, associated to studies having collected biosamples, and being part of a network
- search all studies having collected biosamples and having variables about alcohol, and being part of a network
- …

2.3 Associations

The following diagram describes the various documents that can be published in the Mica web portal. Each of them can be edited individually in the Mica Web Application administration interface (except variables, defined in the Opal servers).



2.4 Annotations

Individual studies can be annotated in two ways:

- Manually annotated by a taxonomy vocabulary.
- Inferred annotations from their associated variables.

Manual annotations by vocabulary are primarily for studies that do not have any associated variables but need to be classified by a taxonomy vocabulary.

2.5 Permissions

Three types of permissions can be granted to a user. Each permission is defined by a user role each of which applies different level of restrictions on a document. The table below lists each role and corresponding restrictions:

Role	Description
Reader	Read-only access to the document in draft mode with its revisions and its associated files.
Editor	Edit access to the document in draft mode with its revisions and its associated files. Publication or permanent deletion are not permitted.
Reviewer	Full access to the document, including its publication, permanent deletion and permissions.

2.6 Revision History

The revision history of a document is the succession of states after each edition (state refers to the content of the document, not its status). This history of changes allows to:

- view changes,
- reinstate a revision,
- identify which state is published.

2.7 Comments

To enhance the collaboration between Mica users, each member can add a comment on any Mica domain document as well as data access requests documents. Mica can be configured to send email notifications when a comment is added or updated.

2.8 Cart

The cart groups a set of variables based on one or more search criteria. User can edit the cart by adding or removing variables, download its content (variables) as a CSV file and use it as a search criterion. Variables of a cart can be attached to a Data Access Request.

The cart can also contain studies and networks. Similarly the content of the cart can be downloaded as CSV files (for internal content analysis). Documents of the cart also be compared between each other.

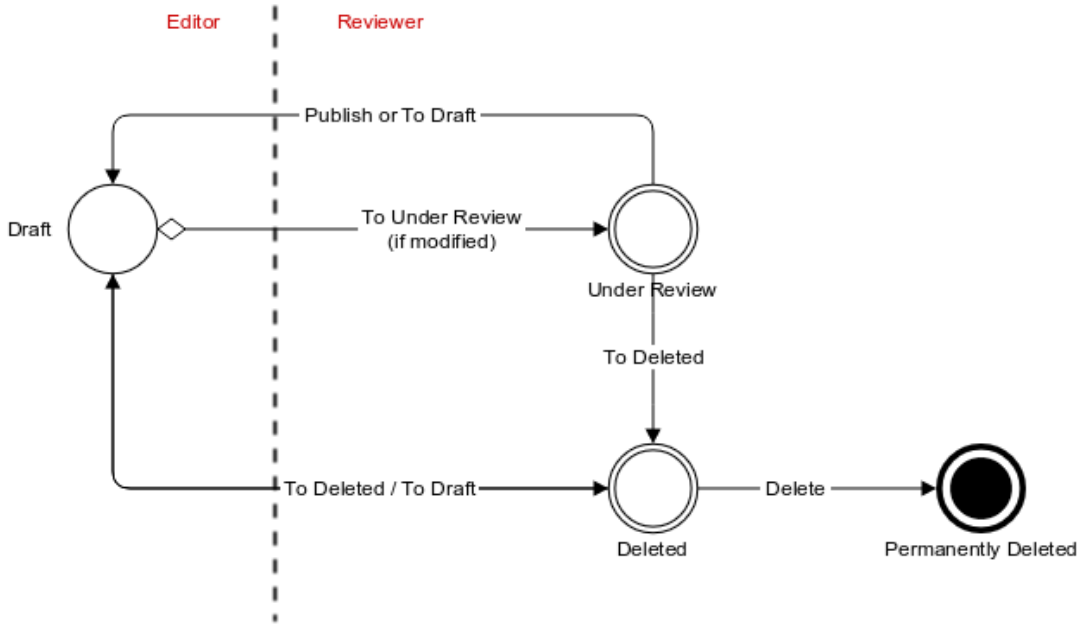
CHAPTER

THREE

PUBLICATION FLOW

Documents (and their associated files) are all publishable documents (except *Data Access Request*). Being a publishable document means that there can be different revisions of the document before being published.

The publication flow refers to the work flow from a draft document to its publication. The following diagram represent the life cycle of a document with its *Revision Status* and *Transitions*:



3.1 Revision Status

The publishable document goes through several states allowing to separate user privileges: some users will be responsible for the content edition only, while other users will be responsible for the reviewing and the publication of the document.

A draft document can be changed/edited as many times as necessary. When the edition work is done, the document is staged for being reviewed. The state of the document that is reviewed is the one that will be published. Once the review and the publication is done, the document is ready again for edition. When a document is to be removed, it is first marked as being deleted (without affecting the publication) before being permanently removed.

The revision status is an enumeration of named states:

Status	Description	Editable	Publishable	Deletable	From Status	To Status
`Draft`	The document is in the editable state. This state requires lesser privileges: the document cannot be published nor deleted, it can only be staged for these operations.	✅	❌	❌	• Under Review • Deleted	• Under Review • Deleted
`Under Review`	Staged for reviewing, allowing user with higher privileges to approve and perform publication. The document is not editable and it can be published. Once published it automatically goes back to the Draft status. If the changes are not approved, the status can be switch to Draft without affecting the publication.	❌	✅	❌	• Draft	• Draft • Deleted
`Deleted`	Staged for permanent deletion. The	❌	❌	✅	• Draft • Under Re-	• Draft

3.2 Transitions

The transitions between the different revision status are the following:

Transition	Description	Permission	From Status	To Status
To Under Review	Once changes have be saved, the document is ready to be reviewed.	• Edit • Review	• Draft	• Under Review
To Draft	If reviewed changes or the deletion are rejected, the document can return to the draft state for edition.	• Edit • Review	• Under Review • Deleted	• Draft
Publish	When changes have been reviewed and approved, the document can be published: the current state of the document is persisted in the publication repository.	• Review	• Under Review	• Draft
To Deleted	Approval for document deletion is requested.	• Edit • Review	• Draft	• Deleted
Delete	Deletion is approved and effective. If the document was published, it is removed from the publication repository.	• Review	• Deleted	

CHAPTER

FOUR

INSTALLATION

Mica is a stand-alone Java server application that requires MongoDB as database engine.

4.1 Requirements

4.1.1 Server Hardware Requirements

Component	Requirement
CPU	Recent server-grade or high-end consumer-grade processor
Disk space	8GB or more.
Memory (RAM)	Minimum: 4GB, Recommended: >4GB

4.1.2 Server Software Requirements

Software	Suggested version	Download link	Usage
Java	8	OpenJDK 8 downloads	Java runtime environment
MongoDB	<= 6.0.x	MongoDB Community downloads	Database engine

While Java is required by Mica server application, MongoDB can be installed on another server.

4.2 Install

Mica is distributed as a Debian/RPM package and as a zip file. The resulting installation has default configuration that makes Mica ready to be used (as soon as a MongoDB server is available). Once installation is done, see *Configuration* instructions.

4.2.1 Debian Package Installation

Mica 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 Mica.
- Manage Mica Service: after package installation, Mica server is running: see how to manage the Service.

4.2.2 RPM Package Installation

Mica 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 Mica.
- Manage Mica Service: after package installation, Mica is running: see how to manage the Service.

4.2.3 Zip Distribution Installation

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

- Download Mica distribution
- Unzip the Mica distribution. Note that the zip file contains a root directory named **mica-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 `MICA_HOME` environment variable
- Separate Mica home from Mica distribution directories (recommended). This will facilitate subsequent upgrades.

Set-up example for Linux:

```
mkdir mica-home
cp -r mica-x-dist/conf mica-home
export MICA_HOME=`pwd`/mica-home
./mica-x-dist/bin/mica
```

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

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

4.2.4 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) would be:

```
version: '3'
services:
      mica:
              image: obiba/mica
              ports:
                      - "8872:8082"
              links:
                      - mongo
```

(continues on next page)

(continued from previous page)

```
                        - agate
                        - opal
                environment:
                        - JAVA_OPTS=-Xmx2G
                        - MICA_ADMINISTRATOR_PASSWORD=${MICA_ADMINISTRATOR_PASSWORD}
                        - MONGO_HOST=mongo
                        - MONGO_PORT=27017
                        - OPAL_URL=http://opal:8080
                        - AGATE_URL=http://agate:8081
                volumes:
                        - /opt/mica:/srv
        mongo:
                image: mongo:6.0
        opal:
                image: obiba/opal
                ports:
                        - "8870:8080"
                links:
                        - mongo
                        - rock
                        - agate
                environment:
                        - OPAL_ADMINISTRATOR_PASSWORD=${OPAL_ADMINISTRATOR_PASSWORD}
                        - MONGO_HOST=mongo
                        - MONGO_PORT=27017
                        - AGATE_URL=http://agate:8081
                        - ROCK_HOSTS=rock:8085
                volumes:
                        - /opt/opal:/srv
        rock:
                image: obiba/rock
        agate:
                image: obiba/agate
                ports:
                        - "8871:8081"
                links:
                        - mongo
                environment:
                        - AGATE_ADMINISTRATOR_PASSWORD=${AGATE_ADMINISTRATOR_PASSWORD}
                        - MONGO_HOST=mongo
                        - MONGO_PORT=27017
                        - RECAPTCHA_SITE_KEY=${RECAPTCHA_SITE_KEY}
                        - RECAPTCHA_SECRET_KEY=${RECAPTCHA_SECRET_KEY}
                volumes:
                        - /opt/agate:/srv
```

Then environment variables that are exposed by this image are:

Environment Variable	Description
JAVA_OPTS	
MICA_ADMINISTRATOR_PASSWORD	Mica administrator password, required and set at first start.
MONGO_HOST	MongoDB server host.
MONGO_PORT	MongoDB server port, default is 27017.
MONGO_DB	MongoDB database nale, default is mica.
OPAL_URL	Opal server URL (optional).
OPAL_HOST	[Deprecated, use OPAL_URL] Opal server host (optional).
OPAL_PORT	[Deprecated, use OPAL_URL] Opal server port, default is 8443.
AGATE_URL	Agate server URL.
AGATE_HOST	[Deprecated, use AGATE_URL] Agate server host.
AGATE_PORT	[Deprecated, use AGATE_URL] Agate server port, default is 8444.

4.3 Upgrade

The upgrade procedures are handled by the application itself.

4.3.1 Debian Package Upgrade

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

```
apt-get install mica
```

4.3.2 RPM Package Upgrade

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

```
yum install mica
```

4.3.3 Zip Distribution Upgrade

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

4.4 Execution

4.4.1 Server launch

Service

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

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

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

```
service mica help
```

The Mica service log files are located in **/var/log/mica** directory.

Manually

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

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

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

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

```
mica
```

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

The Mica server log files are located in **MICA_HOME/logs** directory. If the logs directory does not exist, it will be created by Mica.

4.4.2 Usage

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

- http://localhost:8082 will provide a connection without encryption,
- https://localhost:8445 will provide a connection secured with ssl.

4.4.3 Troubleshooting

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

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

CHAPTER

FIVE

CONFIGURATION

The file **MICA_HOME/conf/application.yml** is to be edited to match your server needs. This file is written in YAML format allowing to specify a hierarchy within the configuration keys. The YAML format uses indentations to express the different levels of this hierarchy. The file is already pre-filled with default values (to be modified to match your configuration), just be aware that you should not modify the indentations. In the following documentation, the configuration keys will be presented using the dot-notation (levels are separated by dots) for readability.

5.1 HTTP Server Configuration

Mica server is a web application and as such, you need to specify on which ports the web server should listen to incoming requests.

Property	Description
`server.port`	HTTP port number. Generally speaking this port should not be exposed to the web. Use the https port instead.
`server.host`	Web server host name.
`server.context-path`	The URL's context path, starting with a */*. For instance when setting */repo*, the base URL will be *https://example.org/repo*.
`https.port`	HTTPS port number.

5.2 MongoDB Server Configuration

Mica server will store its data (system configuration, networks, studies, datasets, etc.) in a MongoDB database. You must specify how to connect to this database.

Property	Description
`spring.data.mongodb.uri`	MongoDB URI. Read Standard Connection String Format to learn more.

By default MongoDB does not require any user name, it is highly recommended to configure the database with a user. This can be done by enabling the Client Access Control procedure.

Follow these steps to enable the Client Access Control on your server:

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

Note: Once the MongoDB service runs with Client Access Control enabled, all database connections require authentication.

MongoDB User Creation Example

The example below creates the *micaadmin* user for *mica* database:

```
use admin

db.createUser( {
  user: "micaadmin", pwd: "micaadmin",
  roles: [
       { "role" : "readWrite", "db" : "mica" },
       { "role" : "dbAdmin", "db" : "mica" },
       { "role" : "readAnyDatabase", "db": "admin" }
  ]
});
```

Here is the required configuration snippet in **/etc/mica/application.yml** for the above user:

```
spring:
  data:
    mongodb:
      uri: mongodb://micaadmin:micaadmin@localhost:27017/mica?authSource=admin
```

Note: Mica 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.

5.3 Opal Server Configuration

Mica server uses Opal to retrieve data dictionaries, data summaries and variable taxonomies. This server is sometimes referred as the Opal primary server (secondary servers can be defined at study level). If you want to publish datasets, the following Opal connection details needs to be configured.

Property	Description
`opal.url`	Opal server URL. It is highly recommended to use https protocol.
`opal.username`	User name for connection to Opal server.
`opal.password`	User password for connection to Opal server.
`opal.token`	Opal personal access token for connection to Opal server. If defined it has the priority over the username/password credentials.

Mica server should connect to Opal and access to some selected tables only with the lowest level of permissions (View dictionary and summary, i.e. no access to individual data). Please refer to the Opal Table Documentation for more details about the permissions that can be applied on a table.

5.4 Agate Server Configuration

Mica server uses Agate as a user directory and as a notification emails service. From the Agate point of view, Mica is not a user: it is an application. Each time Mica needs a service from Agate, it will provide the information necessary to its identification. The application credentials registered in Mica are to be specified in this section. If you want to specify advanced permissions or allow users to submit data access requests, the following Mica connection details needs to be configured.

Property	Description
`agate.url`	Agate server URL. It is highly recommended to use https protocol.
`agate.application.name`	Mica application ID for connection to Agate server.
`agate.application.key`	Mica application key for connection to Agate server.

5.5 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 Mica 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.

5.6 Shiro Configuration

Shiro is the authentication and authorization framework used by Mica. There is a minimum advanced configuration that can be applied to specify how Shiro will hash the password. In practice this only applies to the users defined in the shiro.ini file. Default configuration is usually enough.

Property	Description
`shiro.password.nbHashIterations`	Number of re-hash operations.
`shiro.password.salt`	Salt to be applied to the hash.

5.7 Elasticsearch Configuration

Mica server embeds Elasticsearch as its search engine. Elasticsearch is a key functionality of Mica as the process of publication consist in indexing documents (networks, studies, variables etc.) in the search engine. Advanced queries can be applied on the published documents. Elasticsearch is embeded, i.e. it is not an external application. Mica's Elasticsearch can be part of a cluster of Elasticsearch cluster. The configuration of the Elasticsearch node and how it should connect to the other nodes of the cluster can be specified in this section. Default configuration is usually enough.

Property	Description
`elasticsearch.dataNode`	Boolean to specify if this node has data or if it is just a proxy to other nodes in a cluster.
`elasticsearch.clusterName`	Cluster identifier.
`elasticsearch.shards`	Number of shards.
`elasticsearch.replicas`	Number of replicas.
`elasticsearch.settings`	A string in JSON or YAML format to define other elasticsearch settings. See Elasticsearch Documentation for advanced settings.
`elasticsearch.transportClient`	Boolean to indicate to use the Transport Client instead of creating an elasticsearch Node.
`elasticsearch.transportAddress`	Elasticsearch service IP address and port when using the Transport Client, defaults to the localhost at port 9300.
`elasticsearch.transportSniff`	Boolean to indicate the Transport Client to collect IP addresses from nodes in an elasticsearch cluster.
`elasticsearch.maxConcurrentJoinQueries`	Maximum count of ES queries that can be executed concurrently. Default value is `4`.
`elasticsearch.concurrentJoinQueriesWaitTimeout`	Wait timeout when executing concurrent ES queries in millis. Default value is `30000` milliseconds.

Elasticsearch Cluster

Mica can be set to join or connect to an Elasticsearch cluster. You need to set *elasticsearch.clusterName* to the name of the cluster you want to join. There are different possible cluster topologies, each of which has different resource utilization profiles in terms or memory and CPU.

Note: To avoid API incompatibility issues, the recommended version of Elasticsearch server is 2.4.

An example of a configuration to join an elasticsearch cluster using a Client Node:

```
elasticsearch:
  clusterName: mycluster
  dataNode: false
  settings: '{"node.master": false, "node.local": false}'
```

An example of a configuration using the transport client:

```
elasticsearch:
  clusterName: mycluster
  transportClient: true
  transportAddress: "myhost:9300"
```

Elasticsearch Server Configuration

Mica uses the scripting capabilities of Elasticsearch. All the machines in the Elasticsearch cluster should have the scripting module enabled by setting the following values in the *elasticsearch.yml* configuration file (location of this file depends on how your elasticsearch service is installed):

```
script:
  inline: true
  indexed: true
```

5.8 Miscelaneous Configuration

Advanced settings.

Property	Description
`dar.reminder.cron`	Schedule (cron syntax) of the email reminder for Data Access report. Default value is `0 0 0 * * ?` (every day at midnight).
`sets.cleanup.cron`	Schedule (cron syntax) of the expired cart/sets cleanup. Default value is `0 0 * * * ?` (every hour).

5.9 User Directories

The security framework that is used by Mica for authentication, authorization etc. is Shiro. Configuring Shiro for Mica is done via the file **MICA_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 Mica Debian/RPM package).

By default Mica 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 user directory provided by Agate.

Although it is possible to register some additional user directories, this practice is not recommended as Agate provides more than a service of authentication (user profile, notification emails etc.).

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

File Based User Directory

The file-based user directory configuration file **MICA_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 and a password for the anonymous 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]
```

(continues on next page)

(continued from previous page)

```
# 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/mica2/tools && ./shiro-hasher -p
#   UNIX:
#     cd <MICA_DIST_HOME>/tools && ./shiro-hasher -p
#   WINDOWS:
#     cd <MICA_DIST_HOME>/tools && shiro-hasher.bat -p
#
# Format is:
# username=password[,role]*
administrator = $shiro1$SHA-256$500000$dxucP0IgyO99rdL0Ltj1Qg==$qssS60kTC7TqE61/JFrX/
↪OEk0jsZbYXjiGhR7/t+XNY=,mica-administrator
anonymous = $shiro1$SHA-256$500000$dxucP0IgyO99rdL0Ltj1Qg==$qssS60kTC7TqE61/JFrX/
↪OEk0jsZbYXjiGhR7/t+XNY=

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

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

```
cd /usr/share/mica2/tools
./shiro-hasher -p
```

5.10 Reverse Proxy Configuration

Mica 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 mica.your-organization.org
    ProxyRequests Off
    ProxyPreserveHost On
    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>
```

(continues on next page)

(continued from previous page)

```
    RewriteEngine on
    ReWriteCond %{SERVER_PORT} !^443$
    RewriteRule ^/(.*) https://mica.your-organization.org:443/$1 [NC,R,L]
</VirtualHost>
<VirtualHost *:443>
    ServerName mica.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:8445/
    ProxyPassReverse / https://localhost:8445/
</VirtualHost>
```

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

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

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"
```

CHAPTER

SIX

PUBLIC PAGES CONFIGURATION

Starting from Mica 4.0, the administration user interface is distinct from the public pages, i.e. pages that are to be accessed by regular users. These pages are based on templates that can be customized, extended or overridden. The template engine that is used is FreeMarker which has a clean and powerful syntax.

6.1 Page Templates

6.1.1 Configuring Pages

The main public pages are:

Page	Description
`index`	The home page
`profile`	The user profile page for updating personal information and password
`signin`	The login page
`signup`	The user registration page
`signup-with`	The user registration page, with form pre-filled with personal information extracted from a Op
`forgot-password`	The page to ask for password reset
`just-registered`	The welcome page after a user has registered
`networks`	The list of networks
`network`	The network page
`studies`	The list of studies and initiatives
`study`	The study page (can be individual study or harmonization initiative)
`datasets`	The list of datasets and protocols
`dataset`	The dataset page (can be collected datasets or harmonization protocols)
`variable`	The variable page (can be collected, data schema or harmonized)
`search`	The catalog search page
`projects`	The list of approved projects
`project`	The approved project page
`data-access-process`	The data access process presentation page
`data-accesses`	The list of data access requests (restricted access)
`data-access`	The data access request main page (there are other pages for each of the data access request for
`data-access-agreement-form`	The data access request agreement form page
`data-access-amendment-form`	The data access request amendment form page
`data-access-comments`	The data access request comments page
`data-access-documents`	The data access request documents page
`data-access-feasibility-form`	The data access request feasibility form page
`data-access-form`	The data access request main form page

conti

Table 1 – continued from previous page

Page	Description
data-access-history	The data access request history page
data-access-preliminary-form	The data access request preliminary form page
data-access-private-comments	The data access request private comments page
contact	The "Contact Us" form to send a contact request to the administrators or data access officers
cart	The variables/studies/networks cart page

The templates structure is organized in a way that it should not be necessary to override these main pages definitions. Instead of that, it is recommended to change/extend the theme/style as described in this guide.

The process of configuring these pages is the following (by order of priority):

1. Alter default settings

Some template variables (date formats, branding, favicon etc.) are defined in libs/settings.ftl and can be altered in the file **models/settings.ftl** that would be added in your configuration folder as follows:

```
MICA_HOME
└── conf
    └── templates
        └── models
            └── settings.ftl
```

If enough, this is the less intrusive approach. Note that you do not need to redefine all the settings, just reassign the ones of interest.

General settings

Variable	Description
defaultLang	The default language to be used when extraction labels from documents. If no text version is found for the page's language, this default language's version will be looked up.
datetimeFormat	The format in which the date-time values should be rendered.
date	The format in which the date values should be rendered.
faviconPath	The location of the favicon, to be modified to match your own.
brandImageSrc	The location of your organization's logo.
brandImageCSS	CSS classes to apply to the logo.
brandTextEnabled	Enabled to show/hide a text aside of the logo.
brandTextCSS	CSS classes to apply to the text aside of the logo.
networkIcon	CSS classes to apply to the <i> element to render the icon that represents a network.
studyIcon	CSS classes to apply to the <i> element to render the icon that represents a study.
initiativeIcon	CSS classes to apply to the <i> element to render the icon that represents a harmonization initiative.
datasetIcon	CSS classes to apply to the <i> element to render the icon that represents a dataset.
harmoDatasetIcon	CSS classes to apply to the <i> element to render the icon that represents a harmonization dataset.
variableIcon	CSS classes to apply to the <i> element to render the icon that represents a variable.
dataschemaIcon	CSS classes to apply to the <i> element to render the icon that represents a dataschema variable.
projectIcon	CSS classes to apply to the <i> element to render the icon that represents a project.
taxonomyIcon	CSS classes to apply to the <i> element to render the icon that represents a taxonomy.
adminLTEPath	The location of the AdminLTE theme if this one has been modified (see the **Theme** section in this documentation).

Home page settings

Variable	Description
`networksLink`	The link to list the networks. Default is the **Networks** menu.
`studiesLink`	The link to list the individual studies. Default is the **Individual Studies** menu.
`initiativesLink`	The link to list the harmonization initiatives. Default is the **Harmonization Initiatives** menu.
`datasetsLink`	The link to list the collected datasets. Default is the **Collected Datasets** menu.
`protocolsLink`	The link to list the harmonization protocols. Default is the **Harmonization Protocols** menu.
`portalLink`	The link applied to the logo. Default is the data portal itself (same as **Home** menu), but it could also be the organization's main portal.
`showSignin`	Show the link to the Sign in page. Default is **true** when published content is private or data access request submission is enabled or the cart is enabled.

Cart page settings

Variable	Description
`variablesCartEnabled`	Logical to enable the cart of variables. Default is consistent with the application's general configuration, but can be fine-tuned to make the cart visible to users within roles or groups.
`studiesCartEnabled`	Logical to enable the cart of studies. Default is consistent with the application's general configuration, but can be fine-tuned to make the cart visible to users within roles or groups.
`networksCartEnabled`	Logical to enable the cart of networks. Default is consistent with the application's general configuration, but can be fine-tuned to make the cart visible to users within roles or groups.
`cartEnabled`	Logical to show/hide the cart links (**Cart** menu, addition/removal to/from cart buttons). Default is true when one of the variables, studies or networks cart is enabled. It can be fine-tuned to make the cart visible to users within roles or groups.
`listsEnabled`	Logical to show/hide the lists links (**Lists** menu, addition to list buttons). Default is consistent with the application's general configuration, but can be fine-tuned to make the lists visible to users within roles or groups.
`showCartDownload`	Logical to allow downloading the content of the cart. Default is restricted to users with administration-related role.
`showCartViewDownload`	Logical to allow downloading the content of the cart in the format of Opal views (for creating views in Opal from a variable selection). Default is restricted to users with administration-related role.
`defaultCartType`	Initial the cart page will show the specified cart type tab (one of 'variables', 'studies' or 'networks'). Default is not specified, i.e. the first cart type tab will be active.

Compare page settings

Variable	Description
`studiesComparisonEnabled`	Logical to enable the comparison of studies from the search page (no authentication required) or from the cart of studies (authentication required, see `studiesCartEnabled`).
`networksComparisonEnabled`	Logical to enable the comparison of networks from the search page (no authentication required) or from the cart of studies (authentication required, see `studiesCartEnabled`).

Contact Us page settings

Variable	Description
`contactEnabled`	Logical to show/hide the **Contact** menu. Default is **true**, but can be restricted to users within roles or groups.

User Profile page settings

Variable	Description
`showProfileRole`	Logical to show/hide the role to which the user belongs.
`showProfileGroups`	Logical to show/hide the groups to which the user belongs.

Repository list pages settings

Variable	Description
`listDisplays`	Enumerate the different ways of rendering the lists of documents (networks, studies or datasets). Possible values are **lines**, **table** and **cards**. Some can be omitted (at least one is required) and the order matters.
`listDefaultDisplay`	Default display of a list of documents (networks, studies or datasets). Default is **lines**.
`networkListDisplays`	Specify enumeration of the different ways of rendering the lists of networks. Default is the same as specified by `listDisplay`.
`networkListDefaultDisplay`	Default display of a list of the networks. Default is the same as specified by `listDefaultDisplay`.
`studyListDisplays`	Specify enumeration of the different ways of rendering the lists of networks. Default is the same as specified by `listDisplay`.
`studyListDefaultDisplay`	Default display of a list of the studies. Default is the same as specified by `listDefaultDisplay`.
`datasetListDisplays`	Specify enumeration of the different ways of rendering the lists of networks. Default is the same as specified by `listDisplay`.
`datasetListDefaultDisplay`	Default display of a list of the studies. Default is **cards**.

Search page settings

Variable	Description
`defaultSearchState`	The state of the interface when entering the search page. Default is showing the list of studies or the list of variables when there is only one study.
`defaultIndividualSearchState`	The state of the interface when entering the individual search page. Default is showing the list of individual studies or the list of variables when there is only one study.
`defaultHarmonizationSearchState`	The state of the interface when entering the harmonization search page. Default is showing the list of harmonization initiatives or the list of variables when there is only one initiative.
`downloadQueryEnabled`	Logical to show/hide the button for downloading the results of the query. Default is **true**, but can be restricted to users within roles or groups.
`showCopyQuery`	Logical to show/hide the button for copying the query string, that can be used in the R or Python API. Default is restricted to users with administration-related role.

continues on next page

Table 2 – continued from previous page

Variable	Description
`mapName`	Map name to be used in the graphic **geographical-distribution-chart**. Default is **world**, possible values are **world**, **europe**, **north-america**, **south-america**, **asia**, **africa** or **oceania**.
`searchCharts`	Show/hide and order the graphics by specifying their name. Possible values are **geographical-distribution-chart**, **study-design-chart**, **number-participants-chart**, **bio-samples-chart** or **study-start-year-chart**.
`searchVariableDisplay`	Logical to show/hide the list of variables resulting from the search. Default is consistent with the application's general configuration.
`searchDatasetDisplay`	Logical to show/hide the list of datasets resulting from the search. Default is consistent with the application's general configuration.
`searchStudyDisplay`	Logical to show/hide the list of studies resulting from the search. Default is consistent with the application's general configuration.
`searchNetworkDisplay`	Logical to show/hide the list of networks resulting from the search. Default is consistent with the application's general configuration.
`searchVariableColumns`	Show/hide and order the column names for the list of variables. Possible values are **label**, **label+description** (variable label with a tooltip that shows the description), **valueType**, **annotations**, **type**, **study**, **population**, **data-collection-event/dce**, **initiative**, **dataset** or **protocol**. This configuration will be used when on the `/search` path.
`searchVariableColumnsHarmonization`	Same as the `searchVariableColumns` configuration but will be used when on the `/harmonization-search` path.
`searchVariableColumnsIndividual`	Same as the `searchVariableColumns` configuration but will be used when on the `/individual-search` path.
`searchDatasetColumns`	Show/hide and order the column names for the list of datasets. Possible values are **name**, **type**, **networks**, **studies**, **initiatives** or **variables**.
`searchDatasetColumnsHarmonization`	Same as the `searchDatasetColumns` configuration but will be used when on the `/harmonization-search` path.
`searchDatasetColumnsIndividual`	Same as the `searchDatasetColumns` configuration but will be used when on the `/individual-search` path.
`searchStudyColumns`	Show/hide and order the column names for the list of studies. Possible values are **name**, **type**, **study-design**, **data-sources-available**, **participants**, **networks**, **individual** or **harmonization**.
`searchStudyColumnsHarmonization`	Same as the `searchStudyColumns` configuration but will be used when on the `/harmonization-search` path.
`searchStudyColumnsIndividual`	Same as the `searchStudyColumns` configuration but will be used when on the `/individual-search` path.
`searchNetworkColumns`	Show/hide and order the column names for the list of networks. Possible values are **name**, **studies**, **datasets**, **harmonization**, **individual** or **variables**.
`searchNetworkColumnsHarmonization`	Same as the `searchNetworkColumns` configuration but will be used when on the `/harmonization-search` path.
`searchNetworkColumnsIndividual`	Same as the `searchNetworkColumns` configuration but will be used when on the `/individual-search` path.
`searchVariableFields`	List of variable fields to be extracted from search results.
`searchDatasetFields`	List of dataset fields to be extracted from search results.
`searchStudyFields`	List of study fields to be extracted from search results.
`searchNetworkFields`	List of network fields to be extracted from search results.
`searchVariableSortFields`	List of sort fields to be used for sorting the search. Default is to sort by study, dataset, index (i.e. order in the dataset's data dictionary) and name.
`searchDatasetSortFields`	List of sort fields to be used for sorting the search. Default is to sort by study, population, data collection event and acronym.
`searchStudySortFields`	List of study fields to be used for sorting the search. Default is to sort by acronym.
`searchNetworkSortFields`	List of network fields to be used for sorting the search. Default is to sort by acronym.

continues on next page

Table 2 – continued from previous page

Variable	Description
searchCoverageDisplay	Logical to show/hide the **Coverage** search results tab.
searchGraphicsDisplay	Logical to show/hide the **Graphics** search results tab.
searchListDisplay	Logical to show/hide the **List** search results tab.
searchCriteriaMenus	Show/hide the search criteria in the sidebar by specifying their type (possible values are **variable**, **dataset**, **study**, **network**).

Variable page settings

Variable	Description
showHarmonizedVariableSummary	For a variable summary, allow the possibility to display the summary statistics of a specific harmonized variable. Default is **true**.

Data Access pages settings

Variable	Description
dataAccessFormMenusEnabled	Show/hide the menus panel on the side of the data access form. Default is **true**.
dataAccessDashboardEnabled	Show/hide the dashboard panels on the head of the data access pages. Default is **true**.
dataAccessRemainingTimeEnabled	Show/hide the remaining time in the dashboard page when the data access is approved. Applies only when a project end date can be found. Default is **true**.
dataAccessArchiveEnabled	Show/hide the **Archive** button, to users with appropriate permissions and when the data access request is completed. Default is **true**.
showDataAccessEventsInComments	List of event types in comments form type which events are to be included in the life line of the comments, so that users can contextualize the comments with the changes of the data access forms status. Possible values are **request**, **preliminary**, **feasibility**, **amendment** and **agreement**, default is all.

Charts settings

Variable	Description
barChartBackgroundColor	Background color of the chart elements (the bars or the countries for instance).
barChartBorderColor	Border color of the chart elements.
colors	List of colors to be used for a set of chart elements (portions of a pie chart for instance).

Files settings

Variable	Description
`showFiles`	Logical to show/hide the files that are associated to the documents (networks, studies, populations, data collection events, datasets). Default is **true**, but can be restricted to users within roles or groups. Note that the files can themselves require permissions.
`showNetworkFiles`	Logical to show/hide the files that are associated to the networks. Default is the same as what specified by `showFiles`.
`showStudyFiles`	Logical to show/hide the files that are associated to the studies. Default is the same as what specified by `showFiles`.
`showStudyPopulationFiles`	Logical to show/hide the files that are associated to the study populations. Default is the same as what specified by `showFiles`.
`showStudyDCEFiles`	Logical to show/hide the files that are associated to the study data collection events. Default is the same as what specified by `showFiles`.
`showDatasetFiles`	Logical to show/hide the files that are associated to the datasets. Default is the same as what specified by `showFiles`.

Variables classifications charts settings

Variable	Description
`variablesClassificationsTaxonomies`	Comma separated taxonomies to render the charts of variables classifications coverage (count of variables annotated with each vocabulary). Default is **Mlstr_area**. If the list is empty, no chart will be displayed.
`networkVariablesClassificationsTaxonomies`	Enables classifications taxonomies to render the charts of variables classifications coverage in the network page. Default value is `variablesClassificationsTaxonomies`.
`studyVariablesClassificationsTaxonomies`	Enables classifications taxonomies to render the charts of variables classifications coverage in the study page. Default value is `variablesClassificationsTaxonomies`.
`datasetVariablesClassificationsTaxonomies`	Enables classifications taxonomies to render the charts of variables classifications coverage in the dataset page. Default value is `variablesClassificationsTaxonomies`.

2. Override one of the page model templates

The model templates are to be found in the models folder. This allows to alter some portions of the pages, without affecting the general layout.

The override of the template is done by installing a file with same name, at the same relative location in the application's configuration folder.

```
MICA_HOME
└── conf
    └── templates
        └── models
            └── <template name>.ftl
```

This is the preferred approach when a document's model was modified (new fields added/removed to the network, study, dataset etc.).

3. Override the main page templates

These templates are located at the templates' root folder. This gives full control of the page content but may ignore enhancements or break when upgrading the application.

The override of the template is done by installing a file with same name, at the same relative location in the application's configuration folder.

```
MICA_HOME
└── conf
    └── templates
        └── <template name>.ftl
```

6.1.2 Adding Pages

It is possible to add new pages, for providing additional information or guidance to the regular user. This can be done as follows:

- Install a new page templates
- Add a new menu entry

1. Install custom page template

The new template page is to be declared in the configuration folder:

```
MICA_HOME
└── conf
    └── templates
        └── custom.ftl
```

You can check at the provided templates to make your template fit in the site theme and structure. The profile page template could be a good starting point.

FreeMarker will look at its context to resolve variable values. For a custom page the objects available in the context are:

Object	Description
`config`	The Mica configuration
`user`	The user object (if user is logged in)
`roles`	The list of user roles: `mica-administrator`, `mica-reviewer`, `mica-editor`, `mica-data-access-officer` or `mica-user` (if user is logged in)
`query`	The URL query parameters as a map of strings

This custom template page can load any CSS or JS file that might be useful. These files can be served directly by adding them as follows (there are no restrictions regarding the naming and the structure of these files, as soon as they are located in the **static** folder):

```
MICA_HOME
└── conf
    └── static
        ├── custom.css
        └── custom.js
```

The URL of this custom page will be for instance: `https://mica.example.org/page/custom`.

2. Custom menu entry

To link to a custom page (or an external page), some templates can be defined to extend the default menus: left menu can be extended on its right and right menu can be extended on its left. The corresponding templates are:

```
MICA_HOME
└── conf
    └── templates
        └── models
            ├── navbar-menus-left.ftl
            └── navbar-menus-right.ftl
```

Check at the default left and right menus implementation as a reference.

6.2 Theme and Style

6.2.1 Theme

The default theme is the one provided by the excellent AdminLTE framework. It is based on Bootstrap and JQuery. In order to overwrite this default theme, the procedure is the following:

- Build a custom AdminLTE distribution
- Install this custom distribution
- Change the template settings so that pages refer to this custom distribution instead of the default one

1. Build custom AdminLTE

This requires some knowledge in CSS development in a Node.js environment:

- Download AdminLTE source (source code or a released version)
- Reconfigure Sass variables
- Rebuild AdminLTE (see instructions in the README file, contributions section)

2. Install custom AdminLTE

The objective is to have the web server to serve this new set of stylesheet and javascript files. This is achieved by creating the folder **MICA_HOME/conf/static** and copying the AdminLTE custom distribution in that folder. Not all the AdminLTE are needed, only the **dist** and **plugins** ones. The folder tree will look like:

```
MICA_HOME
└── conf
    └── static
        └── admin-lte
            ├── dist
            └── plugins
```

3. Template settings

Now that the custom AdminLTE distribution is installed in the web server environment, this new location must be declared in the page templates. The default templates settings are defined in the libs/settings.ftl template file. See the **adminLTEPath** variable. This variable can be altered by defining a custom **settings.ftl** file as follows:

```
MICA_HOME
└── conf
    └── templates
        └── models
            └── settings.ftl
```

In this custom **settings.ftl** file the new AdminLTE distribution location will be declared:

```
adminLTEPath = "/admin-lte"/>
```

6.2.2 Style

As an alternative to theming, it is also possible to alter the style of the pages by loading your own stylesheet and tweaking the pages' layout using javascript (and JQuery). The procedure is the following:

- Install custom CSS and/or JS files
- Custom the templates to include these new CSS and/or JS assets

1. Install custom CSS/JS

The objective is to have the web server to serve this new set of stylesheet and javascript files. This is achieved by creating the folder **MICA_HOME/conf/static** and copying any CSS/JS files that will be included in the template pages. The folder tree will look like:

```
MICA_HOME
└── conf
    └── static
        ├── custom.css
        └── custom.js
```

2. Custom templates

For the CSS files, the **models/head.ftl** template allows to extend the HTML pages "head" tag content with custom content. For the JS files, the **models/scripts.ftl** template allows to extend the HTML pages "script" tags. The folder tree will look like:

```
MICA_HOME
└── conf
    └── templates
        └── models
            ├── head.ftl
            └── scripts.ftl
```

Where the **head.ftl** template will be:

```
<link rel="stylesheet" href="/custom.css"/>
```

And the **scripts.ftl** template will be:

```
<script src="/custom.js"/>
```

6.3 Translations

The translations are performed in the following order, for a given `locale`:

1. check for the message key in the messages_<locale>.properties (at different locations)
2. check for the message key in the <locale> JSON object as defined the **Administration > Translations** section of the administration interface

For the messages_* properties, the translations can be added/overridden as follows:

```
MICA_HOME
└── conf
    └── translations
        ├── messages_fr.properties
        └── messages_en.properties
```

Note that you can declare only the messages_* properties files that are relevant (locales available from the website) and the content of these files can contain only the translation keys that you want to override.

CHAPTER

SEVEN

PLUGINS

7.1 Repository

Mica plugins available are:

Name	Type	Description	Depends	API
mica-search-es	mica-search	Mica search engine based on Elasticsearch 2.4. Can be used embedded in Mica (default) or configured to connect to an Elasticsearch cluster.	No dependencies	Search Plugin API
mica-search-es7	mica-search	Mica search engine based on Elasticsearch 7.x. To be configured to connect to an Elasticsearch cluster.	Elasticsearch server	Search Plugin API
mica-tables-spss	mica-tables	Read dataset dictionaries from SPSS files. Reference implementation of the *mica-tables* plugin type.	No dependencies	Table Source Plugin API
mica-taxonomies-files	mica-taxonomies	Read taxonomies from local files or from a URL. Reference implementation of the *mica-taxonomies* plugin type.	No dependencies	Taxonomies Plugin API

7.2 Installation

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

7.2.1 Automatic Installation

Because having a search engine is an absolute requirement, Mica server will check at startup that there is a plugin of type `mica-search` and if it's not the case, the latest version of the mica-search-es plugin (that applies to the current Mica 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 Mica start-up will fail and you will need to install the plugin manually.

7.2.2 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 **MICA_HOME/plugins** folder. Note that the plugin folder name does not matter, Mica 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 Mica.

7.3 Configuration

The MICA_HOME/plugins folder contains all the Mica 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:

```
MICA_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.

7.4 Backups

Mica assigns a data folder location to the plugin: **MICA_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

EIGHT

BACKUP AND RESTORE

The Mica system manages data both in the MongoDB database and in its local configuration/data files (as defined by **MICA_HOME**).

The backup steps are:

1. Stop Mica application
2. Backup Mica local files
3. Backup MongoDB

The restore steps are:

1. Restore MongoDB
2. Restore Mica local files
3. Start Mica application

8.1 Mica

Backup

Some configuration and data are stored in local files and must be backup explicitly:

- `conf`, contains the application's configuration files, including instance specific templates and taxonomies.
- `data`, contains the history of changes of the various documents and the local keystore.
- `plugins`, contains the plugins binary libraries and specific configurations for the Mica server.
- [optional] `work`, contains files that can be rebuilt such as the caches and the search index.

Restore

Mica application includes its own internal upgrade mechanism, then when restoring the runtime environment with a more recent version of Mica, the upgrade process will be automatically triggered. Still, it is recommended to look at the release notes on obiba.org to verify if there are any manual upgrade actions: if this is the case, restore the Mica server one version after the other.

8.2 MongoDB

The MongoDB database contains the different Mica documents in there latest version. See official MongoDB documentation about the different methods for MongoDB Backup/Restore.

8.3 Using Docker

When Mica and MongoDB are managed by Docker, you may consider backup and restore containers volumes the Docker's way: see Backup, Restore or Migrate Data Volumes - Docker documentation.

Note: in the Mica Docker container, **MICA_HOME** is `/srv`.

CHAPTER

NINE

WEB INTRODUCTION

The Mica Web Application is the administration web interface of the Mica server. It is NOT the end-user web portal and therefore firewall policies can (or should) be applied to restrict access to administrators or content editors.

See the *Documents* presentation page for a detailed description of the type of documents that can be edited through this web interface.

The following manuals are available:

- *Documents Management*: add, edit, publish documents and associated files
- *Data Access Requests Management*: approve/reject data access requests
- *Administration*: configure server settings and data access requests form

9.1 Requirements

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

CHAPTER

TEN

DOCUMENTS MANAGEMENT

10.1 View, Edit, Publish

10.1.1 Summary

This guide provides a description of the web interface for viewing and managing documents .

10.1.2 Operations

Edit

Users with Editor or Reviewer permission can modify the properties of a document with status Draft. See *Permissions* and *Publication Flow* for more information.

Publish

Users with Reviewer permission can publish a document with status Under Review.

Note: When a collected dataset is published, its associated study must also be published to update its inferred annotations. It is only during this process that list of variable annotations are collected.

Unpublish

Users with Reviewer permission can unpublish an already published document.

Delete

Users with Reviewer permission can delete a document with status Deleted.

Status Change

Users with Editor or Reviewer permission can change the status of a document. See *Publication Flow* for more information.

10.1.3 Network Operations

Manage Contacts

Users with Editor or Reviewer permission can manage (add/remove/edit) the list of contacts.

Manage Investigators

Users with Editor or Reviewer permission can manage the list of investigators.

Manage Studies

Users with Editor or Reviewer permission can manage the list of studies.

Manage Networks

Users with Editor or Reviewer permission can manage the list of networks.

10.1.4 Individual Study Operations

Manage Contacts

Users with Editor or Reviewer permission can manage (add/remove/edit) the list of contacts. Manage InvestigatorsUsers with Editor or Reviewer permission can manage the list of investigators.

Manage Populations

Users with Editor or Reviewer permission can manage the list of populations.

Manage Data Collection Events

Users with Editor or Reviewer permission can manage the list of data collection events of a population.

Manage Annotations

Users with Editor or Reviewer permission can manage the list of annotations. The list of annotations can be customized to include a subset of variable taxonomies. See the Content configurations in the *administration section*

10.1.5 Harmonization Initiative Operations

Manage Contacts

Users with Editor or Reviewer permission can manage (add/remove/edit) the list of contacts.

Manage Investigators

Users with Editor or Reviewer permission can manage the list of investigators.

10.1.6 Collected Dataset Operations

Manage Study Table

Users with Editor or Reviewer permission can manage (add/remove/edit) the associated study table.

10.1.7 Harmonization Protocol Operations

Manage Study Tables

Users with Editor or Reviewer permission can manage (add/remove/edit) the list of study tables. This operation only applies to harmonization protocols having a collection of study tables.

10.2 Revisions

10.2.1 Summary

Each time a document is edited a new history revision is added. The revisions are ordered from the most recent (current) to the oldest. If the document is published, a star indicates which revision is currently online.

10.2.2 Operations

View

Shows a read-only view of the network of the selected revision.

Restore

Restores the selected revision by replacing the current document. This operation is tracked as a new revision.

10.3 Files

10.3.1 Summary

Mica File System is a repository of files associated with all Mica domain documents . Similar to their associated documents, files have a publication flow and history revisions. The publication flow can apply to one or a group of files. Folders can be used to organize and group files into logical hierarchies but do not represent real data and therefore some operations such as searching do not apply to them.

10.3.2 Operations

Status Change

Refer to *Publication Flow* page for details.

Rename

Users with Editor or Reviewer permission on the containing document can rename a File and Folder. Names cannot contain the following characters: `$ / % #`.

Copy

Selected files and folders can be copied and pasted in other folders.

Move

Selected files and folders can cut and pasted in other folders.

10.3.3 File Specific Operations

Upload

A file from the local file system can be uploaded into the selected folder in the Mica file system.

Download

A file from Mica file system can be download into the local file system.

Search

Files can be searched in two ways:

- free-text where the keyword is matched against the file name , type or description .
- predefined searches listed in the search panel.

The predefined searches are all recursive in that the search query matches all files in all folder hierarchies. By toggling the Recursive button a free-text search can be recursive or applied to the current folder.

Type Edition

File types can be considered as tags and are a comma separated list of keywords associated with a file. They can be edited in the file detailpanel.

Description Edition

File description is localized and can be edited in the file detail panel.

10.3.4 Folder Specific Operations

Folder Creation

Folders can added as a single folder (baseline) or in form of a path (baseline/temp).

10.3.5 Batch Operations

Operations such as copy, move and publication flow can be performed in batch mode when they are applied to a group of selected files and/or folders.

10.3.6 Draft Permissions

The following tables describe file and folder draft permissions:

File Permissions

Role	Description
Reader	Can only view and download a file.
Editor	Can only edit, download and change the status of a file.
Reviewer	All operations are permitted.

Folder Permissions

Role	Description
Reader	Can only view and download a folder.
Editor	Can only edit, download, change the status and upload files into a folder.
Reviewer	All operations are permitted.

Note: File permissions are inherited from their parent folder unless specified.

10.3.7 Publication Access

Publication access makes folders and files available to everyone.

Note: Publication access is only available if Mica content is not configured for *open access*.

10.4 Comments

10.4.1 Summary

All Mica domain *documents* can be commented on by all users with the proper permissions. The content can be pure text or in Markdown format.

10.4.2 Operations

Comment

The entered text (markdown) will be associated with the current document.

Preview

A preview of the rendered markdown text is presented.

Edit

The comment text can be updated and previewed.

Delete

Deletes the selected comment.

10.4.3 Permissions

Role	Description
Reader	Can add, edit and delete own comment. Can view comments of others.
Editor	Can add, edit and delete own comment. Can view comments of others.
Reviewer	Can add, edit and delete own comment. Can view comments of others.
Administrator	Can view, add, edit and delete all comments.

10.5 Permissions

10.5.1 Summary

Access to each publishable *documents* can be controlled. There are actually two sets of privileges:

- **permissions** that apply to **draft** documents: only users having a permission on the draft document can see it,
- **accesses** that apply to **published** documents: by default published documents are open access, i.e. anyone (even a anonymous web portal visitor) can see the publications. This setting `Open access` can be changed in the *General*.

10.5.2 Operations

Add Permission

Adding a permission gives a role to a named user or group of users on the draft document. The available roles are:

Role	Description
Reader	Read-only access to the document in draft mode with its revisions and its associated files.
Editor	Edit access to the document in draft mode with its revisions and its associated files. Publication or permanent deletion are not permitted.
Reviewer	Full access to the document, including its publication, permanent deletion and permissions.

Edit Permission

selected permission role can be modified.

Delete Permission

Delete selected permission.

Add Access

This operation is only available if the `Open access` general setting has been disabled. Adding an access gives the right to see the published document to a named user or group of users. As the *Files permissions* on the associated files can be managed independently, when adding an access there is an option for applying same access to all the files (selected by default). Note that user (or group) name * (star) is an alias for *Anyone* (or *Any group*).

Delete Access

Delete selected access.

CHAPTER

ELEVEN

DATA ACCESS REQUESTS MANAGEMENT

11.1 Summary

Before requesting access to data, a researcher needs to register and follow some steps to submit a *data access request*. This access request is composed of at least one main application form to be filled in by the applicant. Other forms can be enabled for more complex procedures, such as preliminary form, feasibility form, amendment form and end-user agreement form. Each of these forms is to be designed by the administrator.

Once a form is filled-in and submitted by the applicant, data access officers will review it before approval or rejection.

Notification emails help with the communication between the applicant and the data access officer during the life cycle of each form.

Collaborators can be associated by the applicant to a request. Collaborators can only contribute to their own end-user agreement form (if enabled).

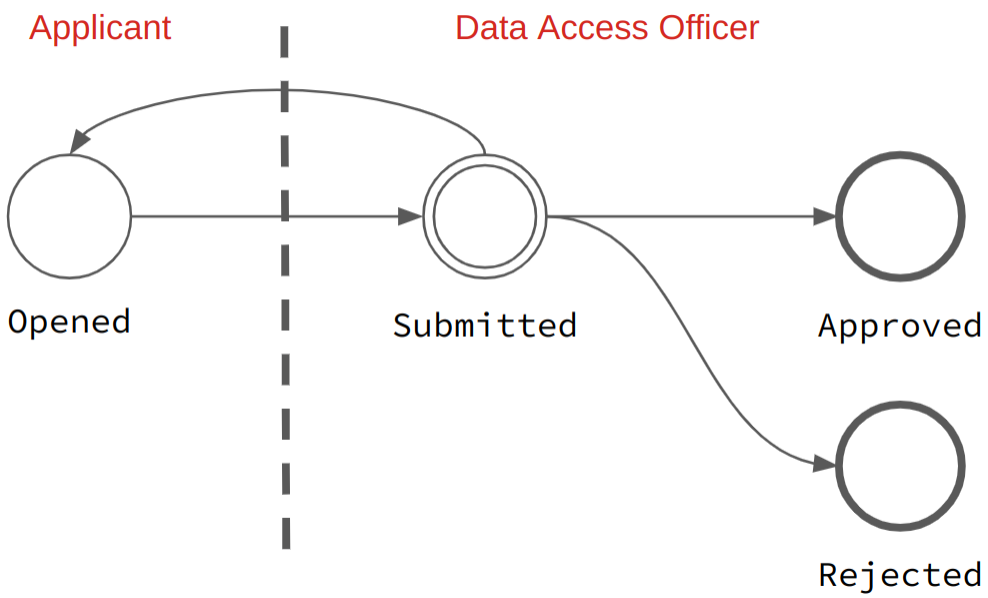
11.2 Request Workflow

To create a data access request, users have to fill out the application form and *submit* it once it is completed. The *validate* button can be used to check that the form contains the required information. Before submitting he request, the form can be edited and saved (for future edits) as needed. After the request is submitted, the form is locked and can no longer be edited unless the request is reopened.

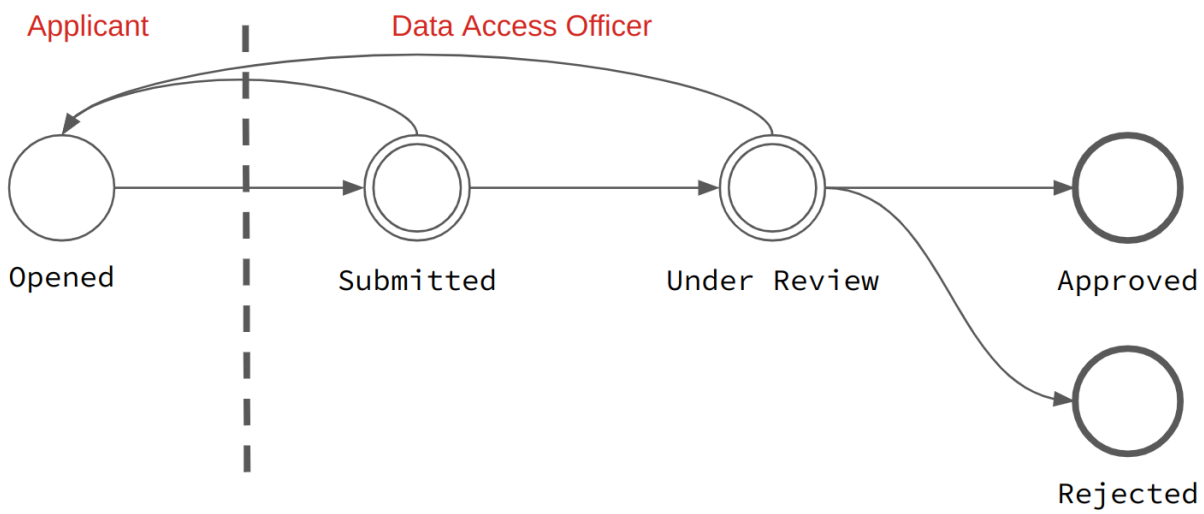
After the request is submitted, a data access officer will *review* it in order to *approve* or *reject* it. The data access officer can also *reopen* (or *conditionally approve*) the request if he considers that modifications are needed. This workflow, as well as email notifications that can be sent on request status changes, can be configured by the administrator, e.g. to omit the *review* step or make the *approve*/*reject* steps final.

The most simple workflow is:

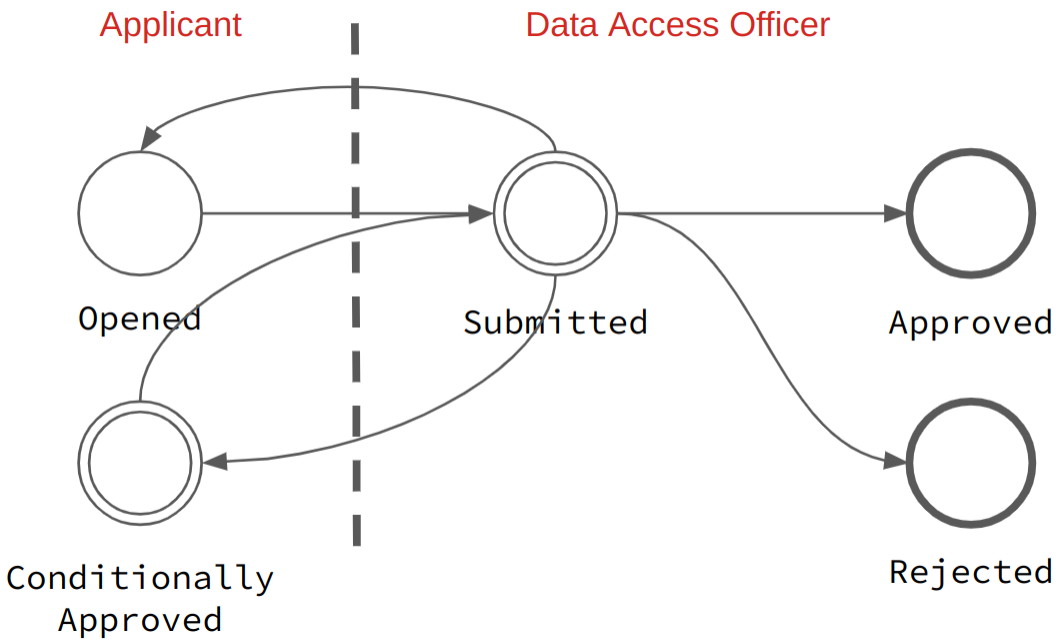
- no `Under Review` intermediate state
- no `Conditionally Approved` intermediate state
- `Approved` state is final
- `Rejected` state is final



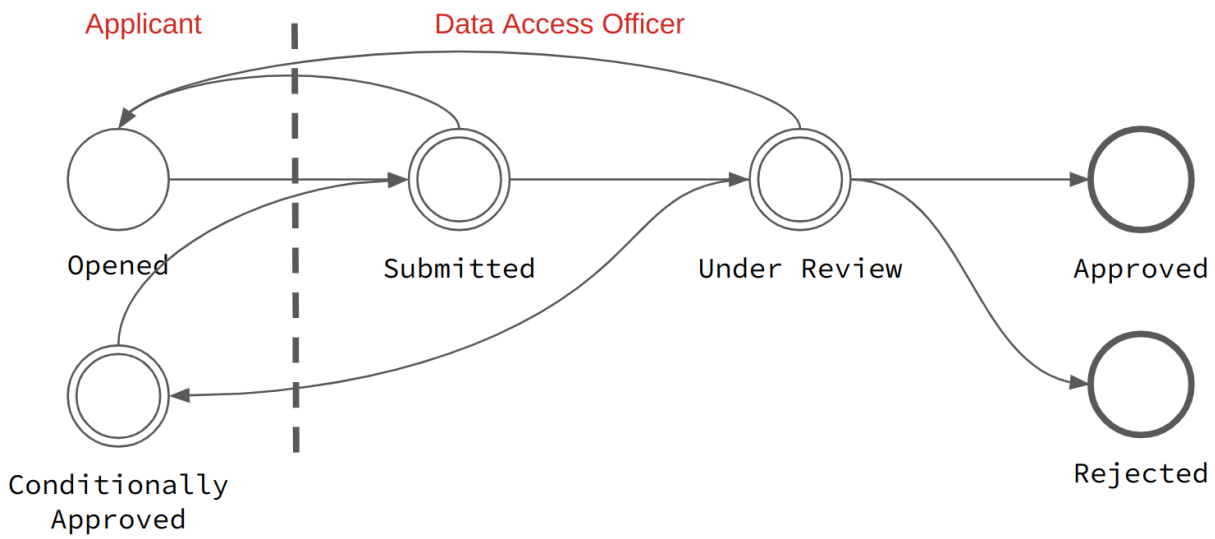
When the `Under Review` intermediate state is activated (which is the default configuration), the flow is:



When the `Conditionally Approved` intermediate state is activated, the flow is:



When both `Under Review` + `Conditionally Approved` intermediate states are activated, the flow is:



The workflow is described in the following table:

Status	Description	From Status	To Status
Opened	The request is in an editable state.		Submitted
Submitted	The request is submitted to the application officer and is not editable by the applicant. The application officer can (conditionally) approve or reject it or, if configured, go to an intermediate under review status.	Opened	Under Review or • Approved • Rejected
Under Review	The request is being reviewed by the application officer, who can reopen, conditionally approve (if this state is activated), approve or reject the request.	Submitted	• Opened • Approved • Rejected • Conditionally Approved • Submitted
Approved	The request has been approved. If configured, the request can go back to the submitted or reviewed status.	Submitted or Under Review	if state is not final: Submitted or Under Review
Rejected	The request has been rejected. If configured, the request can go back to the submitted or reviewed status.	Submitted or Under Review	if state is not final: Submitted or Under Review
Conditionally Approved	The request was submitted, has been reviewed and some adjustments are required before a final approval. The applicant can edit its request and will re-submit it.	Submitted or Under Review	Submitted

11.3 Forms

11.3.1 Application Forms

Preliminary Form (optional)

This form is optional and needs to be designed and enabled by the system administrator.

As setting up a research project can take time in terms of definition of the data/biosamples material needed, organizing the human resources, securing a source of funding etc. the preliminary form allows to start the data access request, before finalizing it with a main form.

Main Form

The application form is to be designed by the system administrator, who can also define the PDF template used to create a printable copy of the form (available by clicking on the *Download* button).

The applicant is responsible for filling-in and submitting this form. The form can be reopened when there are requests for changes. A data access request is approved (resp. rejected) when this form is approved (resp. rejected).

11.3.2 Feasibility Inquiry Form (optional)

This form is optional and needs to be designed and enabled by the system administrator.

Researchers can fill in a short form to ensure the feasibility of a project. This can be a preliminary of the main application form or of an amendment. Several feasibility forms can be submitted.

11.3.3 Amendment Form (optional)

This form is optional and needs to be designed and enabled by the system administrator.

Researchers can submit amendments to request changes (e.g. additional data, new research collaborators) to a pre-approved main form. Several amendment forms can be submitted.

11.3.4 End-User Agreement Form (optional)

This form is optional and needs to be designed and enabled by the system administrator.

In a data access request, the possible strategies to make the end-user agreements available are:

- Always, i.e. as soon as the data access request is created,
- After the Preliminary form is approved,
- After the main form is approved.

When some collaborators are associated to a data access request, each of them must fill-in its own end-user agreement form (the applicant has its own as well).

11.4 Documents

Files can be uploaded by the applicant or the data access officers at any time in the process of the request.

11.5 Comments

To enhance the collaboration between users and data access officer, each member can add a comment to a data access request. Mica can be configured to send email notifications when a comment is added or updated.

11.6 Private Comments

Administrators, data access officers and users with proper permissions (see *data access request permissions*) can add private comments while processing a data access request or an amendment.

11.7 History

A list of all status changes of a data access request and its amendments. In addition, administrators and data access officers can log actions that they have taken while processing a request or an amendment. These actions are either *pre-defined* or free text. Only users or groups with proper permissions (see *data access request permissions*) can view action logs.

11.8 Collaborators

Other users can be invited to collaborate to a data access request. The applicant sends the invitation by providing the collaborator's email address. This collaborator receives a notification email containing a personal link to accept this invitation. This personal link expires after a configurable number of days (7 by default) and is revocable by the applicant. The collaborator must create an account (with the same email) before accepting the invitation.

A collaborator gets read-only access to the request. When the end-user agreement feature is enabled, each collaborator must fill-in and submit its personal end-user agreement statement.

11.9 Notification Emails

For each event in the data access request life-cycle, a notification event can be sent:

Event	Description
On creation	Email is sent to data access officers on data access request creation (i.e. saved for the first time but not yet submitted).
On submission	Email is sent to data access officers and to applicant on data access request submission.
On review	Email is sent to applicant when its data access request is being reviewed.
On reopening	Email is sent to applicant when its data access request is reopened.
On conditional approval	Email is sent to applicant when its data access request is in the 'Conditionally Approved' state.
On rejection	Email is sent to applicant when its data access request is rejected.
On approval	Email is sent to applicant when its data access request is approved.
On comment	Email is sent to applicant and data access officers when the data access request is commented.
On documents update	Email is sent to applicant and data access officers when the document list of the data access request is updated.

There are also notification emails related to the project reporting:

Event	Description
Final report reminder	Email is sent to applicant to remind that a final report is due. Applies to approved data access requests only.
Intermediate report reminder	Email is sent to applicant to remind that an intermediate (yearly) report is due. Applies to approved data access requests only.

Some emails are sent in relation with collaborators:

Event	Description
On collaboration invitation	Email is sent to the invited collaborator with a personal link to the data access request.
On collaboration accepted	Email is sent to applicant when an invitation to collaborate to the data access request has been accepted.

CHAPTER

TWELVE

ADMINISTRATION

The *Administration* menu is available to users with the role `mica-administrator`. This menu gives access to server configuration and status listed below:

12.1 System

12.1.1 General

Summary

The *Administration* menu is available to users with the role `mica-administrator`. This menu gives access to server configuration and status.

Properties

This section allows to define all general Mica configuration as *Server Identification*, *Content*, *Data Source* and *Documents Sections*.

Section	Name	Description
Server Identification	Name	The name of the organization using this instance of Mica server. It will be used when sending notification emails.
	Public URL	Public base URL of the server. It will be used when sending notification emails.
	Portal URL	Public base URL of the portal that will be used when sharing drafts.
Content	Languages	The languages in which the data pertaining to studies, networks and/or datasets will be entered in Mica.
	Default Character Set	The character set with which the data will be entered in Mica e.g. , UTF-8, iso-8859-1, etc.
	Open access	If checked, access to published documents will be opened to everyone.
	Taxonomies for Annotation by Vocabulary	The list of variable taxonomies used in individual study annotation by vocabulary.
Data Source	Primary Opal server Public URL	This Opal server is the primary source of variables and data summaries. (see below)
	Participant Privacy Threshold	No data summary will be returned from Opal if the number of participants is below this threshold.
Application Scope	Single study enabled	If checked, only one study is present in Mica
	Network section enabled	If checked, n etwork section is accessible.
	Single network enabled	If checked, only one network is present in Mica.
	Collected datasets section enabled	If checked, collected datasets section is accessible.
	Harmonization protocols section enabled	If checked, harmonization protocols section is accessible.

To edit a field, click on "Edit" and edit or modify the content the fields therein.

Note: Terminology

By the *primary* Opal server , we mean the Opal server on which Mica looks for data if no other Opal is specified in a study definition. Initially, the primary Opal server is set in Mica Configuration Files , but that entry is overridden by what you enter in here.

Note: Notifications

Mica requires Agate to send email notifications, which should be configured in Mica Configuration Files .

Membership Roles

Networks and Studies can have a list of members associated with them, which are grouped in roles. The list of available roles can be edited and by default the roles contact and investigator are present.

Note: Removing a role from the system is permanent and affects all networks and studies in Mica, i.e. removing and role and recreating it again with the same identifier, won't recover the list of members for that role. However you can still recover the list of members if you revert to an old revision where the role was defined in the system.

Encryption keys

This section presents the tool related to the encryption–through HTTPS–of transactions between Mica and its clients by means of a trusted or a self-signed certificate.

Note: In the instruction below, when you are told to cut and paste the content of the certificate, private key or of an *.pem file, make sure that you copy **all** content, that is* **including** the lines containing " ——BEGIN XXXXXXXX——" and " ——END XXXXXXXX—— ".

Create Key Pair

Go to **Administration > Encryption Keys**:

- Click on the *Add Keys* drop-down.
- Select *Create Key Pair*.
- Fill in the form and click on *Save*.
- **Click on the *Download Certificate* button under the section title** *Encryption Keys*.

Your certificate (*.pem file) should automatically be downloaded on your computer.

Import Key Pair

Go to **Administration > Encryption Keys**:, click on the *Add Keys* drop-down and select *Import Key Pair*.

Here you may use (1) certificate and (2) private key that you got from your institution's IT department. Note that:

- Both the certificate and the private key must in PEM format i.e., you can read them and the file starts with a ——BEGIN [...].
- You must copy the certificate (or the content of the *.crt file) in the public key box and the private key (or the content of the *.key file) in the private key box.

In either case, you finish by clicking on *Save*. Finally, in order for the changes to be taken in account you need to restart Mica with

```
sudo systemctl restart mica2
```

Opals Credentials

In order to establish a secured connection with an Opal server, you must create a user in Opal along with the proper permissions, tell Mica to communicate with that Opal using this user. To do so, there are various scenarios available: you may connect to Opal by means of an SSL certificate or via password/token authentication, these methods are explained in the following three sub-sections. Finally, the last section is about the permission of the user you created in Opal.

Note: In any scenario and for security reasons, never let Mica connect to an Opal as Opal's administrator. You must configure a specific user with appropriate reading permissions.

In **Administration > Opal Credentials** When you click on the drop-down menu *Add Opal Credentials* under the subsection title "Opal Credentials", you are presented with three choices, each corresponding to one of the next three subsections.

Create Key Pair

With this first option, you can create a certificate directly in Mica with which you can create a user in Opal. In order to proceed that way:

- Select "Create" in the drop down menu *Add Opal Credential*.
- **Fill in the necessary information to create the certificate and click on** "Save".
- **The Opal you described at point 2 should now appear in the list under the** *Add Opal Credential* drop-down. At the end of the line for that Opal, click on the download button in the Action column to download the *.pem file which is the certificate created taking in account the information you entered for that Opal and which will be use to add a user with certificate below.

  Note: The URL for that Opal must begin with https:// if we are about to use a certificate as the authentication method.

- Login Opal and go to **Administration > Data Access > Users and Groups**.
- **Click on the drop-down menu Add a User and select the option "Add a user** with certificate…".
- Fill in the info and paste in the content of the *.pem file.
- Save the information.

The user should now be in the list. You'll be done after restarting Mica with

```
sudo systemctl restart mica2
```

Import Key Pair

In the case that you have already have a pair of keys, you may import it here to secure the communication with Opal. You may select "Import" and:

- Fill in the fields (Opal's URL, public and private keys) appropriately.

  Note: Restrictions on how to fill the public key and private key fields using *.pem , *.crt and *.key files are the same as in **Encryption Keys > Import a Certificate** above.

- **You can now proceed as in the instruction to Create a Certificate starting** from point 4.

The user should now be in the list and you'll be done after restarting Mica server.

User Name

This option is probably the easiest:

In Opal:

- Go in **Opal Administration > Data Access > Users and Groups**
- Click on the drop-down menu *Add a User* and select the option "Add a user with password…".
- and you create a user filling the form.

In Mica:

With that user's credentials, i.e. username and password, you select the "User Name" item in the "Add Opal Credential" button. You fill in the form using Opal's URL and the credentials of the user you created in Opal.

Personal Access Token

This is the recommended option as Personal Access Token are safer:

In Opal:

- Go in **Opal Administration > Data Access > Users and Groups**
- Click on the drop-down menu *Add a User* and select the option "Add a user with password…".
- and you create a user filling the form.
- Logout
- Login as the newly created user,
- Click on the user name in the top right corner and select *My Profile* menu OR from the Dashboard page click on the *My Profile* link,
- Select *Add Access Token* and *Add Custom Token…* menu
- In the form, give the token a name, provide which *Projects* can be accessed (optional) and in *Project data access* select *Read-only, without individual-level data*,
- Copy the token.

See also the Opal documentation for making these operations using the User command line or the opalr R package.

In Mica:

You select the "Personal Access Token" item in the "Add Opal Credential" button. You fill in the form using Opal's URL and the token of the user you created in Opal.

Give Permissions to Mica in Opal

You must now give the Mica user the proper permissions on tables in Opal so that the server can carry out his tasks. Here are the steps to do so:

In Opal:

- In **Project > <some specific project> > <some specific table of that project>**
- Click on the "Permissions" tab
- Click on the "Add Permission" button and on "Add user permission" in the drop-down menu

- In the pop-up window, add the name of the user to which you want to grant access and select "View dictionaries and summaries" permission
- Click on save
- Repeat steps for any other table you want the user to have access to

See also the Opal documentation for applying these permissions in bulk using a Table permissions command line or the opalr R package.

12.1.2 Notifications

Summary

When a document/file goes throw a publication flow , relevant users and groups are notified with a status change according to the permissions the y have on the document/file. A reviewer is notified when a document/file status changes to Under review or to Deleted since only a reviewer can publish or permanently delete a document/file. When a document/file status changes to Draft, the editor is notified in order to make the necessary changes on the draft.

Configuration

Notifications can be configured by checking one or many of the following options.

On network status changed notification	Send email notifications when network status changes to Deleted, to Under review or to Draft.
On individual study or harmonization initiative status changed notification	Send email notifications when individual study or harmonization initiative status changes to Deleted, to Under review or to Draft.
On collected dataset status changed notification	Send email notifications when collected dataset status changes to Deleted, to Under review or to Draft.
On harmonization protocol status changed notification	Send email notifications when harmonization protocol status changes to Deleted, to Under review or to Draft.
On file status changed notification	Send email notifications when files status changes to Deleted, to Under review or to Draft.
On comment notification	Send email notifications when a comment is added or updated.
On research project status changed notification	Send email notifications when research project status changes to Deleted, to Under review or to Draft.

12.1.3 Style

The Mica default style of the **administration interface** can be override with a personalized stylesheet. The CSS classes are the ones of Bootstrap (Bootswatch Flatly theme).

See the *Public Pages Configuration* documentation for changing the theme of the published content, the search page, the data access request pages and much more.

12.1.4 Translations

Summary

The Mica's interface can be translated in any language defined in the General Administration. Note that after a language has been added and translations done in the different documents, the content of the site needs to be re-indexed.

Definition

A translation is a key-Value pair that can be added, edited or specified for all configured languages. A translation list is composed of build-in translations and custom translations added through the interface. Note that Mica is already translated in French and English through the build-in translation stored as an application config file.

Operations

Adding translations can be done one key at a time (Add Entry) or by importing a translation file (Import Translation).

Add Entry

First, choose a target language by clicking on a language tab. Then, click on "Add Translation" then on "Add entry" to display the New Translation Entry form. Fill in the Name (key) and Value (Translation) and click on Ok. Scroll down and click on "Save", a new translation key is added. This key should be specified in all the other languages (cf. the "Search/Edit a translation" section).

Import Translation

Instead of being created once at a time, custom translations can be added to the application in one action. Click on "Add Translation" then on "Import Translations", click on "choose file" to choose a translation file (in JSON format). Once the file is uploaded, click on Import.

12.1.5 Caching

Summary

The Administration menu is available to users with the role `mica-administrator`. This menu gives access to server configuration and status.

Definitions

All available caches are listed in the first table under Type . If one or many of these caches becomes desynchronized or do not behave as expected, you may trash the content of the cache so that it will have to be rebuilt. This is done using the trash can icon under the title Action .

There is also an option to build the cache for Dataset variables statistics by clicking the play icon at the right. This specific cache is, by far, the biggest of those used by Mica and may take seconds to hours to build depending on the size of the dataset and various other factors. Having a control over the build is convenient since, for instance, one may want to disconnect Opal from Mica, since the cache is built from the data contained in Opal, the cache has to be functional before the disconnection so that it may continue to work.

12.1.6 Indexing

Summary

The search engine is based on indexed documents. Each index can be rebuilt without affecting the history of changes and the publication status of the documents.

Definitions

Type	Description
All	Indices about any type of document: network, study, dataset (and associated variables), file, person.
Networks	Index that allows to search for networks.
Individual Studies / Harmonization Initiatives	Index that allows to search for individual studies and harmonization initiatives.
Collected Datasets / Harmonization Protocols	Index that allows to search for collected datasets and harmonization protocols and their associated variables.
Collected Datasets	Index that allows to search for collected datasets and associated variables.
Harmonization Protocols	Index that allows to search for harmonization protocols and associated variables.
Persons	Index that allows to search persons who can be associated to studies, initiatives or networks.
Files	Index that allows to search files associated with all Mica documents.
Projects	Index that allows to search projects.
Taxonomies	Index that allows to search for taxonomy terms (i.e. the search criteria).

Note: In order to keep the inferred annotations of individual studies up-to-date, the order of indexing must be as follows:

1. Index the collected datasets which will index their associated vatiables as well.
2. Index the individual studies to update the inferred variable annotations.

12.1.7 Application Metrics

This page provides a many metrics which intend to help the system administrator in his tasks. Information about HTTP requests, service statistics and Ehcache statistics are provided. In addition, one can find information on the JVM concerning memory usage (total, heap and non-heap), garbage collection and threads.

12.1.8 Logs

This is to select what should appear in the log file typically located at `/var/log/mica2`. For each namespace, there are various level you can choose from such as: `trace` , `debug` , `info` , `warn` , and `error`. Each level is typically more informative than the next one.

12.2 Content

12.2.1 Document Types

Summary

This section provides a description of the web interface for configuring the document types. The document type fields configuration is based on the schemaform framework that allows to define the form to collect the model of the document.

Operations

Form

Document fields configuration consists of providing the necessary information (in JSON format) to build a form:

- **Schema**: specifies the name and data type to be collected
- **Definition**: specifies how to layout the form (field positions, translations, section titles, help text)
- **Preview**: is the result of the interpretation of the *schema* and the *definition* by schemaform
- **Model**: displays the data collected (in the preview) according to the specified schema.

Note: Due to the structure of the study type, the form of the study is split in several pieces:

- study: general definition of the study
- population: each study can have one or more populations, this form applies to these only
- and in case of individual studies, data collection event: each population can have one or more data collection events, this form applies to these only.

For detailed documentation on how to use schemaform, see the schemaform documentation. The default schema and definition provided by Mica can also be a good starting point for getting into *schemaform* configuration.

Note that not all fields of a document type are configurable: there are some built-in fields such as name, description… that are necessary for Mica to operate. These fields will appear at the head of the form (when editing a document, not when having a preview of the form configuration). In addition to that other built-in fields are not handled by schemaform , such as the list of studies of a network, the Opal table(s) associated to a dataset, the persons that are members of a study or a network…

It is currently not possible to dynamically integrate *schemaform* addons to Mica. Please contact us if you have a specific need.

Search

Once the fields have been defined, the document search can be configured so that documents can be found by field value. The purpose of the search configuration is then to define a *taxonomy* specific to the considered document type.

The taxonomy is a classification of existing documents based or their fields. It describes the search criteria used to search documents. A criterion is defined on the document's fields and each criterion can (or not) consists in terms related to the values that a field can take. This leads to two criteria types:

- Criteria without terms based on string fields, which leads to a free text search, or based on numerical fields to search a number range.
- Criteria with terms. The terms can be enumerated values for a string, pre-defined number ranges for a numerical field or the values true/false for a Boolean.

Add Criterion

A user with administrator rights can add a criterion to the existing taxonomy.

`Name` (identifier), `Title`, and `Description` are required fields. Under **Definition**, `Field` is required and represents a fully qualified field name under the document model. Check the `Repeatable` definition if search must be an exact match or contained value. Check the `Localized` definition if the field is multilingual.

Under **Display**, one can choose how and if the criterion will be shown.

Checking `Hidden` will hide the criterion from the search while `Facet` controls whether or not it will be exposed in the list page.

Add Term

A user with administrator rights can add a term to an existing criterion. A term is described by a unique `Name` for string criterion or a range for numerical criterion, a `Title`, a `Description` and a set of `Keywords` useful when building search requests.

Dataset/Protocol Search

Despite the fact that collected datasets and harmonization protocols can have different fields, there is only one taxonomy that applies to both document types. See the className search criterion that allows to discriminate datasets by their specific type.

In addition to that, datasets and protocols in Mica have associated variables that are extracted from Opal. Variable model cannot be configured as it lives in Opal, but variable base taxonomy (i.e. that refers the variable properties) can be adjusted in this section.

Permissions

The permissions that apply to all the documents of the considered type can be specified in this section. See *Permissions* for more information.

12.2.2 Data Access Request

Application Form

The web application form can be specified both in terms of schema and UI definition. See Schema Form documentation for more details.

The *Preview* and *Model* tabs are informational only and can be used to respectively preview the rendered form and the input data that will be collected.

Properties

Title Field	Specifies the field which value will be used as the project's title.
Summary Field	Specifies the field which value will be used as the project's summary.
End Date Field	Specifies the field which value will be used to build the project timeline, with an effect on the intermediate and final report notifications.

PDF Download

The data collected in the form can be downloaded in the following formats:

- **PDF Template**, this option requires uploading a template (one for each configured languages) compatible with the custom *Schema Form* model.
- **Printable Page**, this option renders the form as a printable page in the browser so the user can either save the form as a PDF or send it to a printer.

Preliminary Form

Same as the *Application Form*, it uses *Schema Form* to define the schema and UI definition.

Feasibility Form

Same as the *Application Form*, it uses *Schema Form* to define the schema and UI definition.

Amendment Form

Same as the *Application Form*, it uses *Schema Form* to define the schema and UI definition.

Under the **Properties** section one can specify the document's *Title Field* and it's *Summary Field* as defined in the custom *Schema Form*.

End User Agreement Form

Same as the *Application Form*, it uses *Schema Form* to define the schema and UI definition.

Notifications

Email notifications can be sent, if configured, to applicant and data access officers when an event happens on the data access request. Events can be: status changes, comment additions, or updates. Report notifications can also be configured.

These configurations apply to the data access request as a whole, the main application form, the feasibility forms and the amendments.

Settings

The data access request goes through several steps. Some minimum settings can be applied to control this workflow, i.e. enabling the *review* status and making the accepted and rejected status final. Also the pattern to generate identifiers for data access requests can be configured.

General

Permissions

By default the Data Access Officers (DAOs) do not have the permission to edit the application forms, only the applicant and the administrator can do it. It is possible to set the edit permission for DAOs.

Other Forms

Other forms within a data access request can be enabled:

- Amendments, is the capability of requesting amendments to an approved data access request. This option is disabled by default.
- Feasibility inquiries, is the possibility, at any time of the request lifecycle, to ask for a quick evaluation of a project application or amendment. This option is disabled by default.

Variables

It is possible to attach a set of variables to a form (main, amendment or feasibility forms). This set of variables can be defined by the data access request applicant while browsing the catalog of variables, or searching for variables: the variables of interest can be set in a cart which can then be associated to a data access request. Once the request is submitted, the set of variables cannot be modified.

Workflow

The data access request lifecycle can be adjusted with different options:

- Under review intermediate state
- Conditionally approved intermediate state
- Approved final state
- Rejected final state

See *Data Access Requests Management* documentation for details.

ID Generation

Rules for generating a data access request unique identifier can be defined:

- ID prefix
- ID length, including leading zeros or not

Predefined Action Names

Predefined Action Names are common actions that *data access officers* perform while processing the data access requests. These actions can then be logged in the data access request *history section*.

The action names defined here are in fact translation keys that must be created in *Translations*.

The following steps demonstrate how these names are added:

- add the action names without any spaces or any of these invalid characters: `.` `~` `!`
- save the configuration
- add the action keys in *Translations*

Note: An action key is the action name preceded by its translation path: `data-access-request.action-log.config.label.<action-name>`.

Permissions

Role	Description
Reader	Can view all data access request and their amendments.

There are also two additional (sub) permissions that can be granted to a *Reader*:

Apply the same permission to associated action logs	This permission grants viewing the action logs.
Give access to the private comments section	This permission grants viewing and editing of private comments.

Customizing Reports

> **Warning:** This is an **experimental** functionality, make sure to backup your database beforehand.

Follow these steps to customize the data access request and amendment reports:

- stop Mica server
- get the current report configuration files as templates:

```
mongo mica --eval 'db.dataAccessForm.find({}, {csvExportFormat: 1, _id: 0})'
mongo mica --eval 'db.dataAccessAmendmentForm.find({}, {csvExportFormat: 1, _id: 0})
↪'
```

- make sure `/etc/mica2/config/data-access-form/` and `/etc/mica2/config/data-access-amendment-form/` directories exist
- copy your templates `export-csv-schema.json` under the previously created directories
- clear the *csvExportFormat* field in dataAccessForm and dataAccessAmendmentForm:

```
mongo mica --eval 'db.dataAccessForm.update({_id: "default"}, {$set:
↪{csvExportFormat: ""}})'
mongo mica --eval 'db.dataAccessAmendmentForm.update({_id: "default"}, {$set:
↪{csvExportFormat: ""}})'
```

- edit your templates:
  - `/etc/mica2/config/data-access-form/export-csv-schema.json`
  - `/etc/mica2/config/data-access-amendment-form/export-csv-schema.json`
- to make sure that these files can be accessed by Mica server run the following shell command:

```
``sudo chown -R mica:adm /etc/mica2``
```

- start Mica server

The snippet below shows a report configuration file:

```
{
  "headers": {
    "title": {
      "en": "<Organization> Access Office",
      "fr": "<Organisation> Bureau d'accès"
    },
    "subtitle": {
      "en": "Access Requests Report",
      "fr": "Rapport sur les demandes d'acces"
    },

  },
  "table": {
    "generic.accessRequestId": {
      "en": "ACCESS REQUEST ID",
      "fr": "ID DE LA DEMANDE D'ACCES"
```

(continues on next page)

(continued from previous page)

```
    },
    "projectTitle": {
      "en": "TITLE",
      "fr": "TITRE"
    },

  }
}
```

Where fields under `headers` are fixed (built-in) but their translations can be modified. Fields under `table` can be fully customized (removed, re-ordered, added, etc).

The `table` properties can be inferred from the document's schema. They are the fields found in the model.

Note: Properties prefixed by *generic.* are internal and not part of the data access request or amendment form schemas and are considered *built-ins*. They can be removed, however.

12.3 Data Discovery

12.3.1 Building Queries

Summary

The search module allows users to filter and explore Mica documents by building taxonomy based queries. There are several types of criteria where each criterion is associated with one vocabulary. Below are some of the different types of criteria:

- Matching Criterion is used on vocabularies of type string.
- In Criterion are used on vocabularies of type string with several terms each term describing a possible value.
- Range Criterion are used on vocabularies of numeric type having several terms each term describing a bounding tuple [from, to).
- Numeric Criterion are used on vocabularies of numeric type without terms in a given range [from, to).

There are four main taxonomies each of which is used for searching a corresponding Mica document:

- Variable taxonomies
- Dataset taxonomy
- Study taxonomy
- Network taxonomy

The Variable taxonomies is a set of three specific taxonomies, two of which are described and imported from Opal and are developed by Maelstrom Research to allow annotating study and harmonized variables:

- **Areas of information**: classification developed by Maelstrom Research
- **Source & Target**: information about the collected variable

The only variable related taxonomy defined in Mica is **Variable properties** that is used to describe the Opal variable attributes.

Operations

Create a search criterion

Below are the three possible ways of creating query criteria:

Search box

The search box is a quick way of searching taxonomy vocabularies/terms to create a criterion. Typing keywords displays a type-ahead suggesting all matching terms and their corresponding vocabularies in all found taxonomies.

Search properties

The Search properties is a taxonomy specific browser that allows the selection of vocabularies and terms. The left-most column contains the list of all vocabularies, the middle column lists the terms of a selected vocabulary and the last column is the selected term. A criterion can be created by adding a selected vocabulary or one of its terms to the query.

Classifications

The classification section is a more descriptive view of the taxonomies grouped by Mica document types, namely, Variable, Dataset, Study and Network. Similar to Search properties, a criterion can be created either by selecting a vocabulary or one its terms.

12.3.2 Exploring Results

Summary

After building a search query as explained in the Building Queries page, corresponding search result can be visualized in the same search page. Query results are organized in three main tabs: *List*, *Coverage* and *Graphics*.

Operations

List

The List tab displays the results of the search query inventoried in the documents catalogue and organized in four different tabs; Networks, Studies, Datasets and Variables. Each tab describes in details the corresponding search result and contains links that redirect either to a page (e.g variable page, study page, network page) or to a count-search page (e.g variable count, study count).

Coverage

The Coverage displays the results of the search query in a tabular format, giving an overview of which document covers at least one of the search criteria, where the documents are Study, Dataset (Collected Dataset and Harmonization Protocol) and Network. The table reports the total number of variables linked to each document and for each search criterion. Click on any number to obtain the corresponding list of variables. More advanced results can be obtains using the following functionalities.

- **Full coverage**: The study and Dataset list can be filtered by clicking on the "Full Coverage" button to keep only documents that collected information on all search criteria covered by the query. Select documents individually using the check-boxes to the left of the table and click on the button "Filter" the keep a document sub-set.
- **Data Collection Event (DCE)**: Check "Data Collection Event (DCE)" box to obtain the breakdown of variable numbers by data collection events. Click on the button "Full Screen" to better visualize the results. For more information about the Data Collection Event, refer to Mic a documents page, section Study.

Graphics

The tab Graphics summarizes the characteristics of the studies meeting the search criteria in terms of geographical distribution, study design, number of participants and collected biological samples. Click on any count displays the corresponding search page.

Download Results

The search result of any tab can be downloaded and stored in a csv file by clicking on the button Download.

CHAPTER

THIRTEEN

PYTHON INTRODUCTION

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

13.1 Requirements

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

13.2 Installation

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

```
sudo pip install obiba-mica
```

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 mica-python-client

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

Note: This python package depends on the pycurl package which has some system dependencies. One simple solution is to install the `pycurl` system package before.

```
# on Debian systems
sudo apt-get install python3-pycurl

# on RPM systems
sudo yum install python3-pycurl
```

13.3 Usage

To get the options of the command line:

```
mica --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:

```
mica <subcommand> --help
```

This command will display available subcommands.

CHAPTER

FOURTEEN

AUTHORIZATION COMMANDS

Document authorization (on draft and published versions) management.

14.1 Document Access

This command is used to manage the access to a document. This access affects the **published** version and also applies to all associated files in their published version (unless the access to the files is explicitly excluded).

```
mica access-<DOCUMENT> ID <CREDENTIALS> [OPTIONS] [EXTRAS]
```

14.1.1 Arguments

Argument	Description
DOCUMENT	Mica document: `network`, `individual-study`, `harmonization-study`[1], `collected-dataset`, `harmonized-dataset`[2] (see *Documents*)
ID	Identifier of the document

14.1.2 Options

Option	Description
`--add, -a`	Add an access
`--delete, -d`	Delete an access
`--no-file, -nf`	Do not grant access to associated files
`--subject, -s`	Subject name to which the access will be granted
`--type TYPE, -ty TYPE`	Subject type: `user` or `group`

[1] `harmonization-study` refers to `Harmonization Initiative`.
[2] `harmonized-dataset` refers to `Harmonization Protocol`.

14.1.3 Credentials

Authentication is done by username/password credentials.

Option	Description
`--mica MICA, -mk MICA`	Mica server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User password.

14.1.4 Extras

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

14.1.5 Example

Network

Add access for the user demouser on the network demo:

```
mica access-network --mica http://mica-demo.obiba.org --user administrator --password␣
↪password --type USER --subject demouser --add demo
```

Remove the above permission:

```
mica access-network --mica http://mica-demo.obiba.org --user administrator --password␣
↪password --type USER --subject demouser --delete demo
```

Individual Study

Add access for the user demouser on the individual study demo:

```
mica access-individual-study --mica http://mica-demo.obiba.org --user administrator --
↪password password --type USER --subject demouser --add demo
```

Remove the above permission:

```
mica access-individual-study --mica http://mica-demo.obiba.org --user administrator --
↪password password --type USER --subject demouser --delete demo
```

14.2 File Access

This command is used to manage the access to a file in the Mica file system. This access affects the **published** version.

```
mica access-file PATH <CREDENTIALS> [OPTIONS] [EXTRAS]
```

14.2.1 Arguments

Argument	Description
PATH	Path to the file in the Mica file system

14.2.2 Options

Option	Description
--add, -a	Add an access
--delete, -d	Delete an access
--subject, -s	Subject name to which the access will be granted
--type TYPE, -ty TYPE	Subject type: user or group

14.2.3 Credentials

Authentication is done by username/password credentials.

Option	Description
--mica MICA, -mk MICA	Mica server base url.
--user USER, -u USER	User name. User with appropriate permissions is expected depending of the REST resource requested.
--password PASSWORD, -p PASSWORD	User password.

14.2.4 Extras

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

14.2.5 Example

Add access for user demouser on demo individual-study files:

```
mica access-file /individual-study/demo --mica http://mica-demo.obiba.org --user␣
↪administrator --password password --type USER --subject demouser --add
```

Remove the above access:

```
mica access-file /individual-study/demo --mica http://mica-demo.obiba.org --user␣
↪administrator --password password --type USER --subject demouser --delete
```

14.3 Document Permission

This command is used to manage the permissions of a document. These permissions affects the **draft** version and apply to all associated files in their draft version.

```
mica perm-<DOCUMENT> ID <CREDENTIALS> [OPTIONS] [EXTRAS]
```

14.3.1 Arguments

Argument	Description
DOCUMENT	Mica document: `network`, `individual-study`, `harmonization-study`[1], `collected-dataset`, `harmonized-dataset`[2] (see *Documents*)
ID	Identifier of the document

14.3.2 Options

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

[1] `harmonization-study` refers to `Harmonization Initiative`.
[2] `harmonized-dataset` refers to `Harmonization Protocol`.

14.3.3 Credentials

Authentication is done by username/password credentials.

Option	Description
`--mica MICA, -mk MICA`	Mica server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User password.

14.3.4 Extras

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

14.3.5 Example

Network

Add reader permission for the user demouser on the network demo:

```
mica perm-network --mica http://mica-demo.obiba.org --user administrator --password␣
↪password --type USER --subject demouser --add --permission reader demo
```

Remove the above permission:

```
mica perm-network --mica http://mica-demo.obiba.org --user administrator --password␣
↪password --type USER --subject demouser --delete demo
```

Individual Study

Add reader permission for the user demouser on the individual study demo:

```
mica perm-individual-study --mica http://mica-demo.obiba.org --user administrator --
↪password password --type USER --subject demouser --add --permission reader demo
```

Remove the above permission:

```
mica perm-individual-study --mica http://mica-demo.obiba.org --user administrator --
↪password password --type USER --subject demouser --delete demo
```

CHAPTER

FIFTEEN

DOCUMENT COMMANDS

Document management, upload, download, import, publication, search etc.

15.1 Update Collected Dataset

This command is for updating and/or publishing an existing Collected Dataset. The goal is to automate the linkage between a table in Opal with a collected dataset in Mica.

```
mica update-collected-dataset ID <CREDENTIALS> [OPTIONS] [EXTRA]
```

15.1.1 Arguments

Argument	Description
ID	The collected dataset identifier

15.1.2 Options

Option	Description
--study STUDY, -std STUDY	The associated study.
--population POP, -pop POP	The population of the associated study.
--dce DCE, -dce DCE	The data collection event in the population of the associated study.
--project PROJECT, -prj PROJECT	The associated Opal project.
--table TABLE, -tbl TABLE	The table in the associated Opal project.
--publish, -pu	Publish the collected dataset.
--unpublish, -un	Unpublish the collected dataset.

15.1.3 Credentials

Authentication is done by username/password credentials.

Option	Description
`--mica MICA, -mk MICA`	Mica server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User password.

15.1.4 Extras

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

15.1.5 Example

Link a collected dataset in local Mica to a table in Opal.

```
mica update-collected-dataset -u administrator -p password --project CLS --table Wave1␣
↪cls-wave1
```

Associate a collected dataset to a study data collection event in Mica.

```
mica update-collected-dataset -u administrator -p password --study cls --population 1 --
↪dce 1 cls-wave1
```

Publish a collected dataset.

```
mica update-collected-dataset -u administrator -p password --publish cls-wave1
```

15.2 Update Collected Datasets

This command is for updating and/or publishing a list Collected Datasets which are ID is filtered by a regular expression. The goal is to automate the linkage between a table in Opal with a collected dataset in Mica.

```
mica update-collected-datasets ID <CREDENTIALS> [OPTIONS] [EXTRA]
```

15.2.1 Arguments

Argument	Description
`ID`	A regular expression to filter the collected dataset identifiers.

15.2.2 Options

Option	Description
`--project PROJECT, -prj PROJECT`	The associated Opal project.
`--dry DRY, -d DRY`	Dry run of the command to list the collected datasets matching the regular expression.
`--publish, -pu`	Publish the collected datasets.
`--unpublish, -un`	Unpublish the collected datasets.

15.2.3 Credentials

Authentication is done by username/password credentials.

Option	Description
`--mica MICA, -mk MICA`	Mica server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User password.

15.2.4 Extras

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

15.2.5 Example

Link the collected datasets which ID starts with 'cls-wave' in local Mica to a project in Opal and publish them.

```
mica update-collected-datasets -u administrator -p password --project CLS --publish '^
↪cls-wave'
```

15.3 File Management

This command is for advanced users wanting to directly access to the File System API of Mica server.

```
mica file PATH <CREDENTIALS> [OPTIONS] [EXTRA]
```

15.3.1 Arguments

Argument	Description
PATH	Path of file or folder in the file system, for instance: /study/foo

15.3.2 Options

Option	Description
--download, -dl	Download file.
--upload UPLOAD, -up UPLOAD	Upload a local file to a folder in Mica file system, requires the folder to be in DRAFT state. If the destination folder does not exist it will be created.
--create CREATE, -c CREATE	Create a folder at a specific location, requires the file to be in DRAFT state.
--copy COPY, -cp COPY	Copy a file to the specified destination folder.
--move MOVE, -mv MOVE	Move a file to the specified destination folder, requires the file to be in DRAFT state.
--delete, -d	Delete a file on Mica file system, requires the file to be in DELETED state.
--name NAME, -n NAME	Rename a file, requires the file to be in DRAFT state.
--status STATUS, -st STATUS	Change file status.
--publish, -pu	Publish a file, requires the file to be in UNDER_REVIEW state.
--unpublish, -un	Unpublish a file.

15.3.3 Credentials

Authentication is done by username/password credentials.

Option	Description
--mica MICA, -mk MICA	Mica server base url.
--user USER, -u USER	User name. User with appropriate permissions is expected depending of the REST resource requested.
--password PASSWORD, -p PASSWORD	User password.

15.3.4 Extras

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

15.3.5 Example

Get the JSON representation of file /study/foo/bar.pdf

```
mica file /study/foo/bar.pdf -mk https://mica-demo.obiba.org -u administrator -p␣
↪password -j
```

Download file /study/foo/bar.pdf

```
mica file /study/foo/bar.pdf -mk https://mica-demo.obiba.org -u administrator -p␣
↪password --download > bar.pdf
```

Upload a file to /study/foo

```
mica file /study/foo -mk https://mica-demo.obiba.org -u administrator -p password --
↪upload ~/bar.pdf
```

Change status and publish file /study/foo/bar.pdf

```
mica file /study/foo/bar.pdf -mk https://mica-demo.obiba.org -u administrator -p␣
↪password --status UNDER_REVIEW
mica file /study/foo/bar.pdf -mk https://mica-demo.obiba.org -u administrator -p␣
↪password --publish
```

15.4 Search

This command allows to extract published information from the search API of Mica server. The output is in CSV format.

```
mica search <CREDENTIALS> [OPTIONS] [EXTRA]
```

15.4.1 Options

Option	Description
`--target TARGET, -t TARGET`	The type of document to be listed: variable, dataset, study, population, dce (data collection event) or network.
`--query QUERY, -q QUERY`	The search query, in RQL (Resource Query Language), that can be copied from the search page. If not specified, no filter is applied.
`--start START, -s START`	Start search at document position (default is 0).
`--limit LIMIT, -lm LIMIT`	Max number of documents to be listed (default is 100).
`--locale LOCALE, -lc LOCALE`	The language of the labels (default is 'en').
`--out OUT, -o OUT`	Output file path. If not specified, result is printed on the console.

15.4.2 Credentials

Authentication is done by username/password credentials.

Option	Description
`--mica MICA, -mk MICA`	Mica server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User password.

15.4.3 Extras

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

15.4.4 Example

Get 1000 published variables.

```
mica search -mk https://mica-demo.obiba.org -u anonymous -p password --target variable --
↪limit 1000
```

Get 1000 (max) published variables about Alcohol from cohort studies:

```
mica search -mk https://mica-demo.obiba.org -u anonymous -p password --target variable --
↪limit 1000 --query 'variable(in(Mlstr_area.Lifestyle_behaviours,(Alcohol))),
↪study(in(Mica_study.methods-design,cohort_study))'
```

Get the cohort studies having collected data about Alcohol:

```
mica search -mk https://mica-demo.obiba.org -u anonymous -p password --target study --
↪query 'variable(in(Mlstr_area.Lifestyle_behaviours,(Alcohol))),study(in(Mica_study.
↪methods-design,cohort_study))'
```

15.5 Import Zip

This command allows to import a zip-archived file produced by Mica. The result of the import will be the creation or the update of the packaged documents and their attachments.

```
mica import-zip  <CREDENTIALS> [EXTRA] PATH
```

A very useful usage of this command is when a series of associated documents should be imported together. For instance, this command permits to import an individual-study, its network and all its associated collected-datasets. Here is how the documents should be organized into sub-folders and archived such that the import command recognizes it as a valid input:

```
- study
  - individual-study-name
    - network-something.json
    - collected-dataset1.json
    - collected-dataset2.json
    - collected-dataset3.json
    - individual-study-name.json
    - attachments
      - attachment-id1
      - attachment-id2
```

Note: attachment-id is the ID used in the document attachments list in the JSON file, this should not be the filename.

Warning: Use this command with special care to prevent overriding existing documents and breaking associations.

15.5.1 Arguments

Argument	Description
PATH	Path to the zip file or directory that contains zip files to be imported.

15.5.2 Options

Option	Description
--add, -a	Add an access
--delete, -d	Delete an access
--no-file, -nf	Do not grant access to associated files
--subject, -s	Subject name to which the access will be granted
--type TYPE, -ty TYPE	Subject type: user or group

15.5.3 Credentials

Authentication is done by username/password credentials.

Option	Description
--mica MICA, -mk MICA	Mica server base url.
--user USER, -u USER	User name. User with appropriate permissions is expected depending of the REST resource requested.
--password PASSWORD, -p PASSWORD	User password.

15.5.4 Extras

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

15.5.5 Example

Import the file import.zip in Mica server running on localhost with user administrator.

```
mica import-zip -mk https://localhost:8445 -u administrator -p password  /path/to/the/
↪file/import.zip
```

Import all the zip files located in a directory with user editor.

```
mica import-zip -mk https://localhost:8445 -u editor -p password  /path/to/the/zips/
↪directory
```

CHAPTER

SIXTEEN

OTHER COMMANDS

Other commands for advanced users.

16.1 Plugin

Manage system plugins. See *Plugins* documentation.

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

16.1.1 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. 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.

16.1.2 Credentials

Authentication is done by username/password credentials.

Option	Description
`--mica MICA, -mk MICA`	Mica server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User password.

16.1.3 Extras

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

16.1.4 Example

List the plugins that are currently installed:

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

16.2 Web Services

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

```
mica rest ws <CREDENTIALS> [OPTIONS] [EXTRA]
```

16.2.1 Arguments

Argument	Description
`ws`	Web service path, for instance: /user/xxx

16.2.2 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).
`--json, -j`	Pretty JSON formatting of the response.

16.2.3 Credentials

Authentication is done by username/password credentials.

Option	Description
`--mica MICA, -mk MICA`	Mica server base url.
`--user USER, -u USER`	User name. User with appropriate permissions is expected depending of the REST resource requested.
`--password PASSWORD, -p PASSWORD`	User password.

16.2.4 Extras

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

16.2.5 Example

Get all the published studies visible to an anonymous user.

```
mica rest /studies -m GET -mk https://mica-demo.obiba.org -u anonymous -p password -a␣
↪application/json -j
```

Add a new individual study document:

```
mica rest /draft/individual-studies -m POST -u administrator -p password -mk https://
↪mica-demo.obiba.org -ct application/json < patate-study.json
```

Search all files of the draft version of a network:

```
mica rest /draft/files-search/network/some-network -m GET -mk https://mica-demo.obiba.
↪org -u administrator -p password -a application/json -j
```

CHAPTER

SEVENTEEN

R INTRODUCTION

Mica R client, available as a R package, enables data exploration of a Mica server published content.

17.1 Requirements

R 3.6.x or 4.x must be installed on the system. See more about R project.

17.2 Installation

You can install Mica R package from the official CRAN repository:

```
install.packages("micar")
```

CHAPTER

EIGHTEEN

R USAGE

All the R commands that perform search return data frames. The query parameter that can be passed as an argument is the one that can be copied from the search page.

18.1 Connection

```
# Load library
library(micar)

# Open connection
m <- mica.login(url="https://mica-demo.obiba.org")

# Make queries
# ...

# Close connection
mica.logout(m)
```

18.2 Search

A unified search service allows to get any kind of document type, the search criteria can be applied to same or other document types. Only the published documents can be searched.

18.2.1 Networks

Search for published networks.

```
mica.networks(m)
mica.networks(m, query="network(in(Mica_network.studyIds,clsa))")
mica.networks(m, query="variable(in(Mlstr_area.Lifestyle_behaviours,Drugs))", locale="en
↪", from=0, limit=10)
```

18.2.2 Studies

Search for studies, populations and data collection events.

```
mica.studies(m)
mica.studies(m, query="study(in(Mica_study.methods-design,cohort_study))")
mica.studies(m, query="variable(in(Mlstr_area.Lifestyle_behaviours,Drugs))", locale="en",
↪ from=0, limit=10)
mica.study.populations(m)
mica.study.dces(m)
```

18.2.3 Datasets

Search for datasets.

```
mica.datasets(m)
mica.datasets(m, query="dataset(in(Mica_dataset.className,HarmonizationDataset))")
mica.datasets(m, query="variable(in(Mlstr_area.Lifestyle_behaviours,Drugs))")
```

18.2.4 Variables

Search for variables.

```
mica.variables(m)
mica.variables(m, query="variable(in(Mlstr_area.Lifestyle_behaviours,Drugs))")
mica.variables(m, query="dataset(in(Mica_dataset.className,HarmonizationDataset))")
```

18.2.5 Taxonomies

The taxonomies describe the variable annotations and are also used to build the search criteria.

```
# Get taxonomies, vocabularies, terms
mica.taxonomies(m,target="variable")
mica.taxonomies(m,target="variable", query="sex", locale="en", taxonomies = list("Mlstr_
↪area", "Mlstr_additional"))
mica.taxonomies(m,target="study")
mica.vocabularies(m,target="variable", query="cancer", locale = "en")
```

CHAPTER

NINETEEN

REST API INTRODUCTION

The REST API allows to do all necessary operations for managing *Documents* and handling *Publication Flow*. As Mica manages both `draft` and `published` states of each document, there are distinct REST entry points, requiring different level of permissions.

19.1 Authentication

Mica supports Basic authentication, which consists of a HTTP request's header field in the form of `Authorization: Basic <credentials>`, where credentials is the Base64 encoding of user's (or application's) ID and password joined by a single colon `:`.

Using cURL, it is as simple as providing the –user option:

```
curl --user <id>:<password> [...]
```

19.2 Authorization

Authorizations are role based. The built-in roles are:

Role	Description
`mica-administrator`	Can edit/publish data and change system configuration.
`mica-reviewer`	Can edit draft data and publish them.
`mica-editor`	Can edit data draft data.
`mica-data-access-officer`	Can manage data access requests.
`mica-user`	Can view published data.

19.3 Clients

Usage examples provided are based on cURL, a command line client, the Python command line tool (see *Python Introduction*) and the R package micar (published content only).

CHAPTER

TWENTY

DRAFT DOCUMENTS

Management of the documents in their draft mode. See *Publication Flow* for more details.

20.1 Draft Networks

Manage operations on the list of networks in draft mode.

20.1.1 List

GET /draft/networks

List a summary of the networks in draft mode.

This entry point requires *Authentication* of a user with `mica-administrator` or `mica-reviewer` or `mica-editor` role, otherwise an empty list is returned.

Example requests

Using cURL

```
curl --user administrator:password -H "Accept:application/json" https://mica-demo.
↪obiba.org/ws/draft/networks
```

Using the *Web Services* Python command line tool

```
mica rest -mk https://mica-demo.obiba.org -u administrator -p password --method GET␣
↪/draft/networks --json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

[
  {
    "id": "bioshare-eu",
    "acronym": [
      {
        "lang": "en",
        "value": "BioSHaRE-EU"
      }
```

(continues on next page)

(continued from previous page)


```
    ],
    "name": [
      {
        "lang": "en",
        "value": "Biobank Standardisation and Harmonisation for Research Excellence␣
↪in the European Union"
      }
    ],
    "studyIds": [
      "finrisk-2007",
      "ship",
      "lifelines"
    ],
    "permissions": {
      "delete": true,
      "edit": true,
      "publish": true,
      "view": true
    },
    "published": true,
    "timestamps": {
      "created": "2021-04-12T06:38:00.904Z",
      "lastUpdate": "2021-04-12T06:38:04.547Z"
    },
    "obiba.mica.EntityStateDto.networkSummaryState": {
      "permissions": {
        "delete": true,
        "edit": true,
        "publish": true,
        "view": true
      },
      "publicationDate": "2021-04-12T06:38:04.571Z",
      "publishedBy": "administrator",
      "publishedId": "57c9b73f7eec70a4ea471e96c741ddd4c41737e9",
      "publishedTag": "3",
      "revisionStatus": "DRAFT",
      "revisionsAhead": 0
    }
  }
]
```


Query Parameters

- **study** (*string*) – List networks linked to the study with provided identifier.
- **query** (*string*) – RQL search query.
- **from** (*numeric*) – List offset. Default value is `0`.
- **limit** (*numeric*) – List maximum count of documents.
- **sort** (*string*) – Sorting field. Default value is `id`.
- **order** (*string*) – Sorting order. Default value is `asc`.
- **exclude** (*strings*) – Document identifiers to be excluded from the list.

- **filter** (*string*) – Document state filter which possible values are: `ALL`, `PUBLISHED`, `UNDER_REVIEW`, `IN_EDITION`, `TO_DELETE`. Default value is `ALL`.

Response JSON Array of Objects

- **id** (*string*) – The document unique identifier.
- **acronym** (*object*) – The acronym (short name) of the document, as an object that describes a localized string.
- **name** (*object*) – The name of the document, as an object that describes a localized string.
- **studyIds** (*strings*) – The identifiers of the studies that are part of the network.
- **permissions** (*object*) – The different actions that can be performed on this document.
- **published** (*boolean*) – Whether the document is published.
- **timestamps** (*object*) – The date times (format ISO-8601) at which the document was created and updated.
- **obiba.mica.EntityStateDto.networkSummaryState** (*object*) – The publication state of the document.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – `*/*`

Response Headers

- Content-Type – `application/json`

Status Codes

- 200 OK – The documents list to which user has read access rights.
- 500 Internal Server Error – Server error.

20.1.2 Create

`POST /draft/networks`

Create a network in draft mode.

This entry point requires *Authentication* of a user with `mica-administrator` or `mica-reviewer` or `mica-editor` role.

20.1.3 Index

`PUT /draft/networks/_index`

Rebuild both draft and published indices for all networks.

This entry point requires *Authentication* of a user with `mica-administrator` role.

Example requests

Using cURL

```
curl --user administrator:password -H "Accept:application/json" -X PUT https://mica-
↪demo.obiba.org/ws/draft/networks/_index
```

Using the *Web Services* Python command line tool

```
mica rest -mk https://mica-demo.obiba.org -u administrator -p password --method PUT␣
↪/draft/networks/_index --json
```

Request Headers

- Authorization – As described in the *Authentication* section

Status Codes

- 200 OK – The indices rebuild task is scheduled.
- 401 Unauthorized – User does not have the permission to perform this operation.
- 500 Internal Server Error – Server error.

20.2 Draft Network

Manage operation on a single network in draft mode.

20.2.1 Get

GET /draft/network/(string: id)?[key=(string: *key*)]

Get a network in draft mode.

This entry point requires *Authentication* of a user with `mica-administrator` or `mica-reviewer` or `mica-editor` role. Other users may provide an temporary access key.

Example requests

Using cURL

```
curl --user administrator:password -H "Accept:application/json" https://mica-demo.
↪obiba.org/ws/draft/network/bioshare-eu
```

Using the *Web Services* Python command line tool

```
mica rest -mk https://mica-demo.obiba.org -u administrator -p password --method GET␣
↪/draft/network/bioshare-eu --json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "id": "bioshare-eu",
  "name": [
    {
      "lang": "en",
      "value": "Biobank Standardisation and Harmonisation for Research Excellence␣
↪in the European Union"
    }
```

(continues on next page)

(continued from previous page)


```
  ],
  "acronym": [
    {
      "lang": "en",
      "value": "BioSHaRE-EU"
    }
  ],
  "description": [
    {
      "lang": "en",
      "value": "<p> </p>\r\n\r\n<p>BioSHaRE is a consortium of European␣
↪leading biobanks and international researchers from all domains of biobanking␣
↪science. The overall aim of the project is to build upon tools and methods␣
↪available to achieve solutions for researchers to use pooled data from different␣
↪cohort and biobank studies. This, in order to obtain the very large sample sizes␣
↪needed to investigate current questions in multifactorial diseases. This aim is␣
↪achieved through the development of harmonization and standardization tools,␣
↪implementation of these tools and demonstration of their applicability.</p>\r\n\r\
↪n<p> </p>\r\n\r\n<p>As part of its mission, BioSHaRE will document␣
↪information collected by participant biobanks and harmonize, integrate and co-
↪analyse biobank-specific data to answer key research questions on chronic␣
↪diseases.</p>\r\n"
    }
  ],
  "content": "{\"website\":\"http://www.bioshare.eu\",\"maelstromAuthorization\":{\
↪"authorized\":false,\"authorizer\":null}}",
  "studyIds": [
    "finrisk-2007",
    "ship",
    "lifelines"
  ],
  "studySummaries": [],
  "logo": {
    "id": "9e000dc1-564b-4561-92f2-9484fb07054b",
    "fileName": "Bioshare.png",
    "type": "logo",
    "lang": "en",
    "size": 42718,
    "md5": "0c64fd3fb833e28abf8c97e6a8678615",
    "timestamps": {
      "created": "2021-04-12T06:38:04.502Z"
    }
  },
  "memberships": [],
  "permissions": {
    "view": true,
    "edit": true,
    "delete": true,
    "publish": true
  },
  "published": true,
  "timestamps": {
```


(continues on next page)

(continued from previous page)


```
    "created": "2021-04-12T06:38:00.904Z",
    "lastUpdate": "2021-04-12T06:38:04.547Z"
  },
  "obiba.mica.EntityStateDto.state": {
    "publishedTag": "3",
    "revisionsAhead": 0,
    "revisionStatus": "DRAFT",
    "publicationDate": "2021-04-12T06:38:04.571Z",
    "publishedBy": "administrator",
    "publishedId": "57c9b73f7eec70a4ea471e96c741ddd4c41737e9",
    "permissions": {
      "view": true,
      "edit": true,
      "delete": true,
      "publish": true
    }
  }
}
```


Query Parameters

- **key** (*string*) – Optional temporary access key.

Response JSON Object

- **id** (*string*) – The document unique identifier.
- **acronym** (*object*) – The acronym (short name) of the document, as an object that describes a localized string.
- **name** (*object*) – The name of the document, as an object that describes a localized string.
- **description** (*object*) – The description of the document, as an object that describes a localized string.
- **content** (*string*) – The document's model content, as a stringified JSON object.
- **logo** (*object*) – The associated logo file.
- **studyIds** (*strings*) – The identifiers of the studies that are part of the network.
- **studySummaries** (*array*) – The list of the associated studies' summary object.
- **memberships** (*array*) – The list of the associated members object.
- **permissions** (*object*) – The different actions that can be performed on this document.
- **published** (*boolean*) – Whether the document is published.
- **timestamps** (*object*) – The date times (format ISO-8601) at which the document was created and updated.
- **obiba.mica.EntityStateDto.state** (*object*) – The publication state of the document.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – `*/*`

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – The document.
- 401 Unauthorized – Unauthorized access.
- 500 Internal Server Error – Server error.

20.2.2 Upate

PUT /draft/network/(string: id)?[comment=(string: *comment*)]

Update a network.

This entry point requires *Authentication* of a user with mica-administrator or mica-reviewer or mica-editor role.

Example requests

Using cURL

```
curl --user administrator:password -H "Accept:application/json" -X PUT -H "Content-
↪Type: application/json" --data-binary "@network.json" https://mica-demo.obiba.org/
↪ws/draft/network/bioshare-eu
```

Using the *Web Services* Python command line tool

```
mica rest -mk https://mica-demo.obiba.org -u administrator -p password --method PUT␣
↪--content-type "application/json" /draft/network/bioshare-eu < network.json
```

Query Parameters

- **comment** (*string*) – Optional revision comment.

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – */*

Response Headers

- Content-Type – application/json

Status Codes

- 204 No Content – Successful update of the document.
- 401 Unauthorized – Unauthorized access.
- 404 Not Found – The document does not exist.
- 500 Internal Server Error – Server error.

20.2.3 Get Model

GET /draft/network/(string: id)/model

Get the model part of a network.

This entry point requires *Authentication* of a user with `mica-administrator` or `mica-reviewer` or `mica-editor` role.

Example requests

Using cURL

```
curl --user administrator:password -H "Accept:application/json" https://mica-demo.
↪obiba.org/ws/draft/network/bioshare-eu/model
```

Using the *Web Services* Python command line tool

```
mica rest -mk https://mica-demo.obiba.org -u administrator -p password --method GET␣
↪/draft/network/bioshare-eu/model --json
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "website": "http://www.bioshare.eu",
  "maelstromAuthorization": {
    "authorized": false,
    "authorizer": null
  }
}
```

Request Headers

- Authorization – As described in the *Authentication* section
- Accept – `*/*`

Response Headers

- Content-Type – `application/json`

Status Codes

- 200 OK – The document's model content.
- 401 Unauthorized – Unauthorized access.
- 500 Internal Server Error – Server error.

20.2.4 Update Model

PUT /draft/network/(string: id)/model

Update the model part of a network.

This entry point requires *Authentication* of a user with `mica-administrator` or `mica-reviewer` or `mica-editor` role.

Example requests

Using cURL

```
curl --user administrator:password -H "Accept:application/json" -X PUT -H "Content-
↪Type: application/json" --data-binary "@model.json" https://mica-demo.obiba.org/
↪ws/draft/network/bioshare-eu/model
```

Using the *Web Services* Python command line tool

```
mica rest -mk https://mica-demo.obiba.org -u administrator -p password --method PUT␣
↪--content-type "application/json" /draft/network/bioshare-eu/model < model.json
```

20.2.5 Index

PUT /draft/network/(string: id)/_index

Rebuild both draft and published indices for a network.

This entry point requires *Authentication* of a user with `mica-administrator` role.

Example requests

Using cURL

```
curl --user administrator:password -H "Accept:application/json" -X PUT https://mica-
↪demo.obiba.org/ws/draft/network/bioshare-eu/_index
```

Using the *Web Services* Python command line tool

```
mica rest -mk https://mica-demo.obiba.org -u administrator -p password --method PUT␣
↪/draft/network/bioshare-eu/_index --json
```

20.2.6 Update Status

PUT /draft/network/(string: id)/_status?value=(string: *status*)

Update the edition status of a network: `DRAFT` when the document is being edited, `UNDER_REVIEW` when the document is to be reviewed for publication, `DELETED` when the document is marked for permanent removal.

This entry point requires *Authentication* of a user with `mica-administrator` or `mica-reviewer` role.

Example requests

Using cURL

```
curl --user administrator:password -H "Accept:application/json" -X PUT https://mica-
↪demo.obiba.org/ws/draft/network/bioshare-eu/_publish
```

Using the *Web Services* Python command line tool

```
mica rest -mk https://mica-demo.obiba.org -u administrator -p password --method PUT␣
↪/draft/network/bioshare-eu/_publish --json
```

Query Parameters

- **status** (*string*) – The edition status which can be one of `DRAFT`, `UNDER_REVIEW`, `DELETED`.

20.2.7 Publish

PUT /draft/network/(string: id)/_publish

Publish a network.

This entry point requires *Authentication* of a user with `mica-administrator` or `mica-reviewer` role.

Example requests

Using cURL

```
curl --user administrator:password -H "Accept:application/json" -X PUT https://mica-
↪demo.obiba.org/ws/draft/network/bioshare-eu/_publish
```

Using the *Web Services* Python command line tool

```
mica rest -mk https://mica-demo.obiba.org -u administrator -p password --method PUT␣
↪/draft/network/bioshare-eu/_publish --json
```

20.2.8 Unpublish

DELETE /draft/network/(string: id)/_publish

Unpublish a network.

This entry point requires *Authentication* of a user with `mica-administrator` or `mica-reviewer` role.

Example requests

Using cURL

```
curl --user administrator:password -H "Accept:application/json" -X DELETE https://
↪mica-demo.obiba.org/ws/draft/network/bioshare-eu/_publish
```

Using the *Web Services* Python command line tool

```
mica rest -mk https://mica-demo.obiba.org -u administrator -p password --method␣
↪DELETE /draft/network/bioshare-eu/_publish --json
```

20.2.9 Remove

DELETE /draft/network/(string: id)

Remove a network (unpublish it if necessary).

This entry point requires *Authentication* of a user with `mica-administrator` or `mica-reviewer` role.

Example requests

Using cURL

```
curl --user administrator:password -H "Accept:application/json" -X DELETE https://
↪mica-demo.obiba.org/ws/draft/network/bioshare-eu
```

Using the *Web Services* Python command line tool

```
mica rest -mk https://mica-demo.obiba.org -u administrator -p password --method␣
↪DELETE /draft/network/bioshare-eu --json
```

CHAPTER

TWENTYONE

PARTNERS AND FUNDERS

The development of this application was made possible thanks to the support of our partners and funders:

CHAPTER

TWENTYTWO

SUPPORT

Please visit OBiBa support page.

HTTP ROUTING TABLE

/draft

GET /draft/network/(string: id)/model, 118
GET /draft/network/(string: id)?[key=(string:key)], 114
GET /draft/networks, 111
POST /draft/networks, 113
PUT /draft/network/(string: id)/_index, 119
PUT /draft/network/(string: id)/_publish, 120
PUT /draft/network/(string: id)/_status?value=(string:status), 119
PUT /draft/network/(string: id)/model, 119
PUT /draft/network/(string: id)?[comment=(string:comment)], 117
PUT /draft/networks/_index, 113
DELETE /draft/network/(string: id), 121
DELETE /draft/network/(string: id)/_publish, 120