mEDRA

Multilingual European DOI Registration Agency

mEDRA Web Service Client Tool
Installation and Use Manual

Version: 2.4
Last Modified: 30 January 2017

1. Introduction

The mEDRA web service provides an interface to the mEDRA registration services to be used in business to business settings. Currently, it provides access to three different operations: batch upload, DOI registration, and view metadata.

- The batch upload operation is to be used for triggering either the DOI or DOICitations deposit on mEDRA and Crossref systems in an asynchronous mode. It can be used also to query Crossref.
- The registration operation is to be used for triggering the DOI registration procedure in a synchronous mode.
- The view metadata operation is used to retrieve the Onix for DOI metadata associated with a given DOI instance.

In order to communicate service requests and responses to the mEDRA web service, the client tool exploits the Secure Socket Layer interface, which enables the communication protocol to meet the requirements of integrity, secrecy and confidentiality.

2. Application Requirements

The mEDRA web service client tool requires a Java Runtime Environment 1.6 or later to be installed on the same host.

3. Installation Instructions

The mEDRA web service client tool is released and distributed in the form of a zip archive. Hence, a file compression utility compatible with the zip format is required to start the installation.

The mEDRA web service client tool is ready to be used after the distribution package has been decompressed into the designated parent directory. After the installation, the files will be organized according to the following directory structure.

PARENT_DIR
medraWSClient/ main installation directory
lib/ Java software libraries employed by the client tool
config.properties mEDRA client tool configuration properties (to be modified)
medraClient.jar mEDRA client main library
README.md mEDRA client tool description
usage.properties mEDRA client tool options (not to be modified)

Tool Configuration
The config.properties file, which is located in the tool root directory, contains the tool default configuration so as to simplify the typical command line invocation. However, in case of need, any single default setting can be overridden by specifying the appropriate command line argument, as specified in the following section.

The default configuration enables the user to set up the following properties:

- *mEDRA web service account information* (medra.web.service.url, medra.cr.web.service.url, medra.account.username, medra.account.password properties): this group of properties enables the user to specify its own mEDRA account information as well as the access points of the mEDRA web service;
- *metadata transformation label* (metadata.transformation.label property): a metadata transformation label is required whenever metadata records not conforming to the Onix for DOI schema are sent to the mEDRA web service. Any different metadata format will require a different transformation label;
- *metadata content type* (metadata.content.type property): the mEDRA web service accepts metadata specified in SGML and XML formats. In order for the registration process to behave correctly, the appropriate metadata content type has to be specified in the service request;
- *metadata validation* (metadata.validating property): when the metadata sent to the mEDRA web service is a XML instance, the metadata validation property can be set to "true" to provide the immediate validation service feature;
- *debug option* (debug property): set the debug option to "true" to run the mEDRA web service in a verbose mode;
- *crossref request* (crossref.request property): set this option to "true" to send DOI or DOICitations also to Crossref;
- *access mode* (access.mode property): the way Crossref deposit should be done: 01 = asynchronous, 02 = synchronous;
- *language* (language property): the language in which the mEDRA responses are to be sent;
- *proxy usage*: the web service client may be run even behind a proxy. In this case, the proxy usage must be set to "true" (default is "false") and the necessary information (proxy.host,proxy.port,proxy.username and proxy.password) regarding the proxy configuration has to be specified.

The user can refer to the config.properties file for having more information about each single property.

4. Usage Instructions

Once installed and configured, the mEDRA web service client is ready to be used.
The command syntax is

java -jar medraClient.jar ((-u | -cu | -r | -q) <file> | -v <doi>) [-transform <label>] [-validate] [-url <WS-URL>] [-username <name> -password <word>] [-debug] [-crossref] [-amode (01|02)] [-lang (ita|ger|eng)]

You can display it by running the command
java -jar medraClient.jar

The parameters listed above can be divided into 2 categories:
1 – operation type:
-u = asynchronous DOI deposit (upload). This command launches the DOI metadata transfer to the mEDRA DOI registration server, triggering a DOI registration in an asynchronous paradigm. The tool execution ends when the metadata transfer operation has terminated and returns a result with a termination status code depending on the outcome of the data transfer process. The outcome of the registration process will be notified by email or HTTPCallback, as the registration process will be terminated. This operation type has been designed to manage DOI registrations when batch of metadata records are submitted.
-cu = asynchronous DOICitations deposit (upload). This command launches the citations metadata transfer to the mEDRA DOI registration server, triggering a DOICitations registration in asynchronous paradigm. The tool execution ends when the citations metadata transfer operation has terminated and returns a result with a termination status code depending on the outcome of the data transfer process. The outcome of the citations deposit process will be notified by email or HTTPCallback, as the citations deposit process will be terminated. This operation type has been designed to manage DOI citations deposit when batch of citations metadata records are submitted.
-r = synchronous DOI deposit. This command launches the DOI metadata transfer to the mEDRA DOI registration server, triggering a DOI registration in a synchronous paradigm. The tool execution ends when the registration operation has terminated and returns a result with a termination status code depending on the outcome of the registration process. This operation type has been designed to manage DOI registrations of single metadata records.
-q = asynchronous query submission to Crossref. This command launches a query transfer to the mEDRA query service triggering a Query Request towards the Crossref system in an asynchronous paradigm. The tool execution ends when the query metadata transfer operation towards Crossref has terminated and returns a result with a termination status code depending on the outcome of the data transfer process. The outcome of the query process will be notified by email or HTTPCallback, as soon as Crossref will send a response to the query.
-v = DOI record download. Given a DOI, this command retrieves from mEDRA metadata repository the last metadata record which has been submitted for that DOI.

2 – options:
-transform (optional) = type of transformation (e.g. JATS)
-validate (optional) = if present, the web service client will validate the file before sending it to the server
-url (optional) = the web service URL
-username (optional) = the username for authentication
-password (optional) = the password for authentication
-debug (optional) = if present, the web service client will run in debug mode
-crossref (optional) = if present, the DOI, DOICitation or Query will be sent also to Crossref. To be used only with –u, -cu or –q operation.
-amode (optional) = access mode (01 for asynchronous, 02 for synchronous). To be used only with –crossref option.
-lang (optional) = the user language (ita for Italian, ger for German and eng for English). To be used only with –crossref option.

5. TrustStore Instructions

The client uses the default jvm trustStore to check if the SSL certificates are trusted.
If you get an SSL exception, please check if the root of the certification chain is in the default jvm trustStore. To do that, you can launch the command with the option
-Djavax.net.debug=all
For example
java -Djavax.net.debug=all -jar medraClient.jar -u <file>

If you cannot add the root of the certification chain to the jvm default trustStore, you can create a local trustStore and add the root of the certification chain to it, for example by using the following command
keytool -import -file <rootcertificate.crt> -alias <alias> -keystore <keystore> -storepass <storepassword>

After that you can launch the client adding the following system properties:
-Djavax.net.ssl.trustStore=<keystore> -Djavax.net.ssl.trustStorePassword=<storepassword>
For example
java -Djavax.net.ssl.trustStore=<keystore> -Djavax.net.ssl.trustStorePassword=<storepassword> -jar medraClient.jar -u <file>