Kuha2 Documentation

Release 0.x.x

Toni Sissala

Feb 14, 2022

Contents:

Kuha2 is a metadata server that provides descriptive social science research metadata for harvesting via multiple protocols and a growing variety of metadata standards. The software is a collection of applications and consists of three server applications, a client application and a database.

The development was initiated by CESSDA SaW -project, but will continue as an Open Source project lead by FSD.

CHAPTER 1

Links

Source code repositories and issue trackers are hosted in Bitbucket.

- Kuha Document Store Latest release: 0.12.0
- Kuha OAI-PMH Repo Handler Latest release: 0.14.1
- Kuha OSMH Repo Handler Latest release: 0.6.1
- Kuha Client Latest release: 0.10.0
- Kuha Common Latest release: 0.15.1
- Kuha Documentation

Documentation for Kuha2 applications is hosted at ReadTheDocs.

- *Kuha Document Store*
- *Kuha OAI-PMH Repo Handler*
- *Kuha OSMH Repo Handler*
- *Kuha Client*
- *Kuha Common*

CHAPTER 2

License

Kuha2 software components are available under the EUPL.

CHAPTER 3

Installation

Each software component needs to be installed separately. Refer to the *Installation* chapter.

3.1 User guide

3.1.1 Architecture

In a typical usage scenario, the repository owner submits DDI files to the Document Store using Kuha Client. Document Store then stores the metadata into a database. Repository handlers implement different harvesting protocols such as OAI-PMH and query the Document Store accordingly. Only repository handlers should be exposed to external use (that is, to harvesters). If access control or traffic shaping is required, Kuha2 can be deployed behind an API gateway or some other proxy.

Kuha2 components communicate to each other through RESTful APIs and the use of the Document Store is not restricted to DDI. It is possible to bypass Kuha Client and use the Document Store API directly to submit metadata to the Store.

3.1.2 Getting started

Once the *installation* is complete, you may wish to populate Document Store with some example data in order to see how the software works. This guide will demostrate how to populate Document Store with example data. The guide assumes all Kuha2 software components are installed on localhost and using default configuration values.

Also refer to OAI-PMH documentation and OSMH documentation for information about the protocols.

Note: The commands in this guide may change in future versions of Kuha Client. Refer to *Kuha Client documentation* for current commands and configuration parameters.

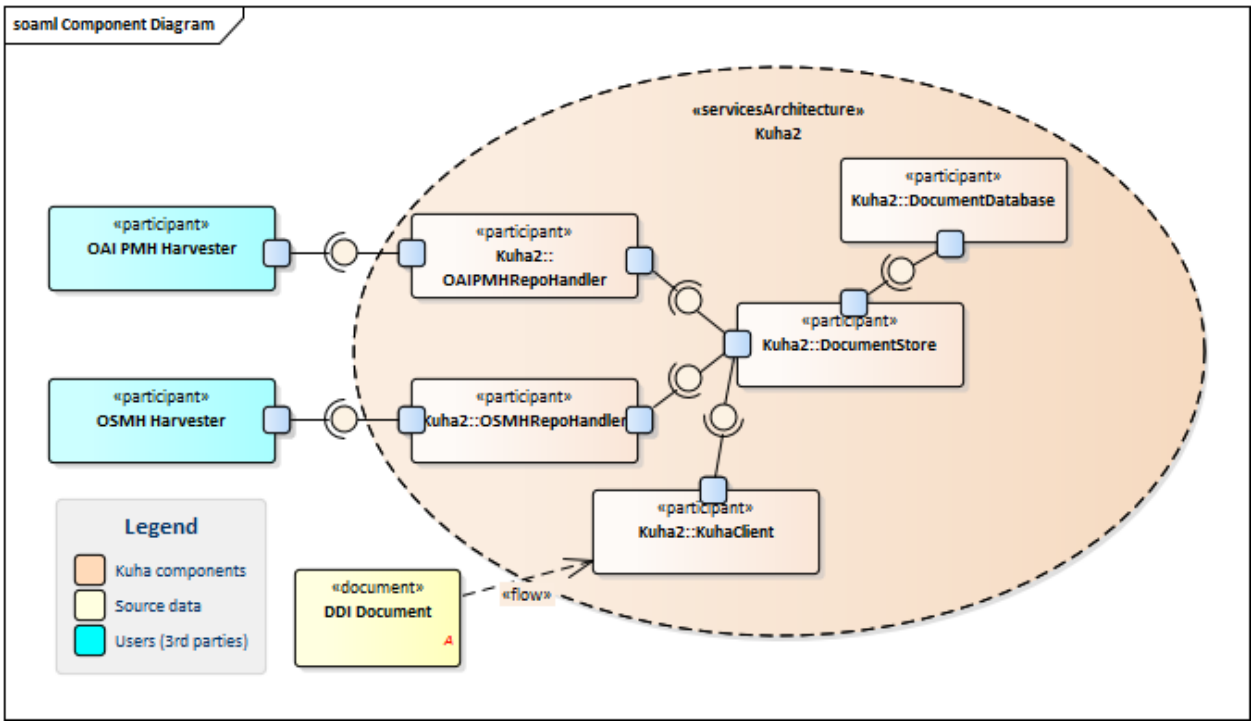

Fig. 3.1: Fig. Kuha2 service architecture diagram.

Importing metadata

Import `Study in DDI 1.2.2.` to create a single Study with study number `study_1` and a single StudyGroup with identifier `serie_1`. The Study and StudyGroup contain some localized content marked with `xml:lang` attributes.

Import the XML metadata to Document Store

```
python -m kuha_client.kuha_import --document-store-url=http://localhost:6001/v0 --
↪source-file-type=ddi_122_nesstar ddi122_minimal_study.xml
```

The metadata is now available via OAI-PMH.

```
curl "http://localhost:6003/v0/oai?verb=GetRecord&metadataPrefix=ddi_c&
↪identifier=study_1"
```

And OSMH.

```
curl "http://localhost:6002/v0/GetRecord/Study/study_1"
curl "http://localhost:6002/v0/GetRecord/StudyGroup/serie_1"
```

`Study in DDI 2.5.` is a similar example, but serialized in DDI 2.5. It creates a single Study with study number `study_2` and a single StudyGroup with identifier `serie_2`. The Study and StudyGroup also contain some localized content.

Import the XML metadata to Document Store.

```
python -m kuha_client.kuha_import --document-store-url=http://localhost:6001/v0 --
↪source-file-type=ddi_c ddi25_minimal_study.xml
```

The metadata is now available via OAI-PMH.

```
curl "http://localhost:6003/v0/oai?verb=GetRecord&metadataPrefix=ddi_c&
↪identifier=study_2"
```

And OSMH.

```
curl "http://localhost:6002/v0/GetRecord/Study/study_2"
curl "http://localhost:6002/v0/GetRecord/StudyGroup/serie_2"
```

`Variables in DDI 2.5.` contains a Study with three Variables and Questions.

Import the metadata.

```
python -m kuha_client.kuha_import --document-store-url=http://localhost:6001/v0 --
↪source-file-type=ddi_c ddi25_minimal_variables.xml
```

See it in OAI-PMH.

```
curl "http://localhost:6003/v0/oai?verb=GetRecord&metadataPrefix=ddi_c&
↪identifier=study_3"
```

And in OSMH.

```
curl "http://localhost:6002/v0/GetRecord/Study/study_3"
curl "http://localhost:6002/v0/GetRecord/Variable/study_3:VAR_1"
curl "http://localhost:6002/v0/GetRecord/Variable/study_3:VAR_2"
curl "http://localhost:6002/v0/GetRecord/Variable/study_3:VAR_3"
curl "http://localhost:6002/v0/GetRecord/Question/study_3:QUESTION_1"
curl "http://localhost:6002/v0/GetRecord/Question/study_3:QUESTION_2"
curl "http://localhost:6002/v0/GetRecord/Question/study_3:QUESTION_3"
```

Updating metadata

To keep the Document Store up-to-date with your DDI metadata, Kuha Client provides a `kuha_upsert` -module. The use is similar to the `kuha_import` module, except that upsert provides an optional command line parameter which instructs the client to remove records that are not found in current run. This is used in batch operations, when you wish to sync a directory full of DDI-files to Document Store.

`Updated Study in DDI 2.5.` has the same study number `study_2`, so it will update the already imported study. This file contains a new distributor (distrbtr-element) and has removed the elements referring to secondary study titles (partitl-elements).

To update the Study to Document Store.

```
python -m kuha_client.kuha_upsert --document-store-url=http://localhost:6001/v0 --
↪source-file-type=ddi_c ddi25_minimal_study_updated.xml
```

See the updated study in OAI-PMH.

```
curl "http://localhost:6003/v0/oai?verb=GetRecord&metadataPrefix=ddi_c&
↪identifier=study_2"
```

And OSMH.

```
curl "http://localhost:6002/v0/GetRecord/Study/study_2"
```

If you run the upsert command with the command line option `--remove-absent`, the other documents imported earlier will be removed, since they are not found from the DDI file.

First assure that the documents imported earlier are still served via OAI-PMH

```
curl "http://localhost:6003/v0/oai?verb=ListIdentifiers&metadataPrefix=ddi_c"
```

Now run the upsert command with `--remove-absent`.

```
python -m kuha_client.kuha_upsert --remove-absent --document-store-url=http://
↪localhost:6001/v0 --source-file-type=ddi_c ddi25_minimal_study_updated.xml
```

Then look at ListIdentifiers again.

```
curl "http://localhost:6003/v0/oai?verb=ListIdentifiers&metadataPrefix=ddi_c"
```

ListIdentifiers should only return the a single `study_2`.

Deleting all records

To remove the example data from Document Store you can issue a delete command with Kuha Client, which will delete all documents from all collections.

```
python -m kuha_client.kuha_delete --document-store-url=http://localhost:6001/v0 ALL␣
↪ALL
```

3.2 Installation

This chapter describes the installation of each application.

3.2.1 Installing Kuha Document Store

This guide will provide step-by-step instructions in installing Kuha Document Store and MongoDB database. Operating system used in this guide is Ubuntu 16.04, but other modern Linux variants may be used.

In this guide the installation of the database is done on a separate server. However, Document Store and MongoDB may be installed on the same server.

Install MongoDB

Note: These actions should be done on the MongoDB server.

It is recommended to use the latest version of MongoDB which can be obtained from MongoDB's own repository. Refer to MongoDB manual on how to install MongoDB to your operating system. At the time of writing the installation to Ubuntu 16.04 was done as follows.

1. Obtain MongoDB public key.

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv␣
↪0C49F3730359A14518585931BC711F9BA15703C6
```

2. Add MongoDB source.

```
echo "deb [ arch=amd64,arm64 ] http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/
↪3.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.4.list
```

3. Update indexes and install.

```
sudo apt-get update && sudo apt-get install -y mongodb-org
```

4. Configure MongoDB to accept incoming connections. **Use IP of your MongoDB server in <mongodb-ip>**.

```
sudo sed -i 's/  bindIp: 127.0.0.1/  bindIp: <mongodb-ip>/' /etc/mongod.conf
```

5. Start MongoDB.

```
sudo service mongod start
```

Now MongoDB is running and accepting incoming connections. Note that the MongoDB instance is now accepting incoming connections without authentication. Authentication will be enabled later in this guide. For up-to-date information on configuration and security see MongoDB manual.

The next step is to create administrator credentials and setup databases, collections and database users. This can be done with a script bundled with Kuha Document Store.

First it is required to install the Document Store package.

Install Document Store

Note: These actions should be done on the Document Store server.

---

1. Create directory for document store and Python virtualenv.

```
mkdir kuha2
```

2. Clone package to subdirectory.

```
git clone --single-branch --branch releases https://bitbucket.org/tietoarkisto/kuha_
↪document_store kuha2/kuha_document_store
```

3. Install Python virtual environment.

```
sudo apt install -y python3-venv
```

4. Make installation script executable.

```
chmod +x ./kuha2/kuha_document_store/scripts/install_kuha_document_store_virtualenv.sh
```

5. Install Kuha Document Store to virtual environment.

```
./kuha2/kuha_document_store/scripts/install_kuha_document_store_virtualenv.sh
```

Upgrade Document Store

In order to upgrade an existing install, fetch changes to code repository, checkout a version and re-install.

1. Change directory to package directory.

```
cd kuha2/kuha_document_store
```

2. Fetch changes and checkout a version to upgrade to.

```
git fetch --all --tags
git checkout <version>
```

3. Leave package directory, make installation script executable and install.

```
cd ../..
chmod +x ./kuha2/kuha_document_store/scripts/install_kuha_document_store_virtualenv.sh
./kuha2/kuha_document_store/scripts/install_kuha_document_store_virtualenv.sh
```

Setup MongoDB for Document Store

Document store provides a script which will help setup MongoDB. The script will prompt for administrator credentials, which will be created. Give hostname/IP of your MongoDB server as command line parameter.

The script will create needed collections and database users. It will also setup indexes for the collections to speed up database queries.

Note: You may wish to provide DB credentials for editor and reader. Give parameter `--help` to see how.

---

1. Make the setup script executable.

```
chmod +x ./kuha2/kuha_document_store/scripts/setup_mongodb.sh
```

2. Run the MongoDB setup script. **Replace <mongodb-ip> with the IP of your MongoDB server.**

```
./kuha2/kuha_document_store/scripts/setup_mongodb.sh --database-host=<mongodb-ip>
```

Now the database is ready to be used with Document Store. Care should be taken to secure the MongoDB instance. For Kuha2 the only IP that needs access to the database is Kuha Document Store's IP. Authentication should also be enabled.

Enable authentication to MongoDB

Note: These actions should be done on the MongoDB server.

---

1. Enable authentication.

```
sudo sed -i 's/#security:/security:\n  authorization: enabled/' /etc/mongod.conf
```

2. Restart MongoDB.

```
sudo service mongod restart
```

Running the Document Store

Note: This action should be done on the Document Store server.

---

1. Make the run-script executable.

```
chmod +x ./kuha2/kuha_document_store/scripts/run_kuha_document_store.sh
```

2. Start serving Document Store. **Replace <mongodb-ip> with the IP of your MongoDB server.**

```
./kuha2/kuha_document_store/scripts/run_kuha_document_store.sh --database-host=
↪<mongodb-ip>
```

3.2.2 Installing Kuha OSMH Repo Handler

The operating system used in these steps is Ubuntu 16.04. Other modern Linux variants may be used.

1. Create directory for OSMH Repo Handler and Python virtualenv.

```
mkdir kuha2
```

2. Clone package to subdirectory.

```
git clone --single-branch --branch releases https://bitbucket.org/tietoarkisto/kuha_
↪osmh_repo_handler kuha2/kuha_osmh_repo_handler
```

3. Install Python virtual environment.

```
sudo apt install -y python3-venv
```

4. Make install script executable.

```
chmod +x ./kuha2/kuha_osmh_repo_handler/scripts/install_kuha_osmh_repo_handler_
↪virtualenv.sh
```

5. Install Kuha OSMH Repo Handler to virtual environment.

```
./kuha2/kuha_osmh_repo_handler/scripts/install_kuha_osmh_repo_handler_virtualenv.sh
```

To run Kuha OSMH Repo Handler you need access to Kuha Document Store. First you will need to make run script executable.

```
chmod +x ./kuha2/kuha_osmh_repo_handler/scripts/run_kuha_osmh_repo_handler.sh
```

Run by calling the script. **Replace <document-store-url> with the URL to the Document Store**.

```
./kuha2/kuha_osmh_repo_handler/scripts/run_kuha_osmh_repo_handler.sh --document-store-
↪url=<document-store-url>
```

Upgrade OSMH Repo Handler

In order to upgrade an existing install, fetch changes to code repository, checkout a version and re-install.

1. Change directory to package directory.

```
cd kuha2/kuha_osmh_repo_handler
```

2. Fetch changes and checkout a version to upgrade to.

```
git fetch --all --tags
git checkout <version>
```

3. Leave package directory, make installation script executable and install.

```
cd ../..
chmod +x ./kuha2/kuha_osmh_repo_handler/scripts/install_kuha_osmh_repo_handler_
↪virtualenv.sh
./kuha2/kuha_osmh_repo_handler/scripts/install_kuha_osmh_repo_handler_virtualenv.sh
```

3.2.3 Installing Kuha OAI-PMH Repo Handler

The operating system used in these steps is Ubuntu 16.04. Other modern Linux variants may be used.

1. Create directory for OAI-PMH Repo Handler and Python virtualenv.

```
mkdir kuha2
```

2. Clone package to subdirectory.

```
git clone --single-branch --branch releases https://bitbucket.org/tietoarkisto/kuha_
↪oai_pmh_repo_handler kuha2/kuha_oai_pmh_repo_handler
```

3. Install Python virtual environment.

```
sudo apt install -y python3-venv
```

4. Make install script executable.

```
chmod +x ./kuha2/kuha_oai_pmh_repo_handler/scripts/install_kuha_oai_pmh_repo_handler_
↪virtualenv.sh
```

5. Install Kuha OAI-PMH Repo Handler to virtual environment.

```
./kuha2/kuha_oai_pmh_repo_handler/scripts/install_kuha_oai_pmh_repo_handler_
↪virtualenv.sh
```

Upgrade OAI-PMH Repo Handler

In order to upgrade an existing install, fetch changes to code repository, checkout a version and re-install.

1. Change directory to package directory.

```
cd kuha2/kuha_oai_pmh_repo_handler
```

2. Fetch changes and checkout a version to upgrade to.

```
git fetch --all --tags
git checkout <version>
```

3. Leave package directory, make installation script executable and install.

```
cd ../..
chmod +x ./kuha2/kuha_oai_pmh_repo_handler/scripts/install_kuha_oai_pmh_repo_handler_
↪virtualenv.sh
./kuha2/kuha_oai_pmh_repo_handler/scripts/install_kuha_oai_pmh_repo_handler_
↪virtualenv.sh
```

To run Kuha OAI-PMH Repo Handler you need access to Kuha Document Store. First make the run script executable.

```
chmod +x ./kuha2/kuha_oai_pmh_repo_handler/scripts/run_kuha_oai_pmh_repo_handler.sh
```

Run by calling the script. **Replace <document-store-url> with the URL to the Document Store**. You also need to specify few configuration values for OAI-PMH: *base_url* and *admin_email*.

```
./kuha2/kuha_oai_pmh_repo_handler/scripts/run_kuha_oai_pmh_repo_handler.sh --document-
↪store-url=<document-store-url> --oai-pmh-base-url=<base_url> --oai-pmh-admin-email=
↪<email>
```

3.2.4 Installing Kuha Client

1. Create directory for Kuha Client and Python virtualenv.

```
mkdir kuha2
```

2. Clone package to subdirectory.

```
git clone --single-branch --branch releases https://bitbucket.org/tietoarkisto/kuha_
↪client kuha2/kuha_client
```

3. Install Python virtual environment.

```
sudo apt install -y python3-venv
```

4. Install Kuha Client to virtual environment

```
cd kuha2
python3 -m venv kuha_client-env
source ./kuha_client-env/bin/activate
cd kuha_client
pip install -r requirements.txt
pip install .
```

Batch import files to Document Store. Python virtual environment must be active.

```
python -m kuha_client.kuha_import --document-store-url=<document-store-url> --file-
↪log-path=file_log /path/to/directory
```

Batch upsert records to Document Store and remove records not in current batch. Python virtual environment must be active.

```
python -m kuha_client.kuha_upsert --document-store-url=<document-store-url> --remove-
↪absent --file-log-path=file_log /path/to/directory
```

Upgrade Kuha Client

In order to upgrade an existing install, fetch changes to code repository, checkout a version and re-install.

1. Change directory to package directory

```
cd kuha2/kuha_client
```

2. Fetch changes and checkout a version to upgrade to

```
git fetch --all --tags
git checkout <version>
```

3. Activate Kuha Client virtual environment

```
source ../kuha_client-env/bin/activate
```

4. Upgrade.

```
pip3 install -r requirements.txt --upgrade --upgrade-strategy=only-if-needed
pip3 install . --upgrade --upgrade-strategy=only-if-needed
```

3.3 Kuha Document Store

Kuha Document Store is a HTTP backend API written in Python for serving Document Store records to multiple repo handlers. The Document Store uses MongoDB as a persistent storage and provides multiple endpoints for managing the database documents.

Kuha Document Store is a part of Open Source software bundle Kuha2.

3.3.1 Features

Import records from DDI XML

Kuha Document Store provides an easy way to import multiple records all at once by simply submitting a DDI file to an import-endpoint. The Document Store imports all records found from the file and handles inserts and updates correctly.

REST API for full control

Kuha Document Store has a REST API that gives end users full control of the records stored in the Document Store. The REST API may be used to build functionality for spesific needs, for example, to automatically update a record when a record is changed in a 3rd party storage system.

With the REST API, end users are not tied to using DDI, but may use arbitrary metadata formats and submit their records to Document Store using HTTP with JSON payload.

Flexible query support

Kuha Document Store provides an endpoint for selectively querying stored records. The Query API is used by client applications which are a part of the Kuha2 software bundle.

3.3.2 Dependencies & requirements

- Python 3.5 or newer
- MongoDB 3.4 or newer (License: GNU AGPL v3.0)
- **Recommended**: python3-venv 3.5.1 or newer

The software is continuously tested against Python versions 3.5, 3.6, 3.7, 3.8 and 3.9.

MongoDB 3.4 is the first supported version, but the software is also known to work with 3.6 and 4.2. Intermediate versions are most likely suitable.

Python packages

The following can be obtained from Python package index.

- motor (License: Apache License 2.0)
- pymongo (License: Apache License 2.0)

- tornado (License: Apache License 2.0)
- Cerberus (License: ICS)
- python-dateutil (License: Simplified BSD)

Kuha Common is a library used with Kuha2 software. It can be obtained from https://bitbucket.org/tietoarkisto/kuha_common

- kuha_common (License: EUPL)

3.3.3 License

Kuha Document Store is available under the EUPL. See LICENSE.txt for the full license.

3.3.4 Configuration

The application can be configured with a configuration file, via command line arguments or by environment variables. All configuration options have default values. If a configuration option is specified in more than one place, then command line values override environment variables which override configuration file values which override defaults.

The following configuration options are available:

-h, --help
Show help message and exit.

--print-configuration
Print active configuration and exit.

--document-store-port <port>
Port of Kuha document store database. Defaults to ´‘6001‘‘. May also be controlled by setting environment variable: KUHA_DS_PORT.

--document-store-api-version <api_version>
Api version for document store. This gets prepended to the URL path. Defaults to v0. May also be controlled by setting environment variable: KUHA_DS_API_VERSION.

--database-host <database_host>
Host/IP of the Document Store database. Defaults to localhost. May also be controlled by setting environment variable: KUHA_DS_DBHOST

--database-port <port>
Port of the Document Store database. Defaults to 27017. May also be controlled by setting environment variable: KUHA_DS_DBPORT

--database-name <name>
Name of Document Store database. Defaults to kuha_document_store. May also be controlled by setting environment variable: KUHA_DS_DBMAME

--database-user-reader <user>
Username for database user having read-only rights. Defaults to reader. May also be controlled by setting environment variable: KUHA_DS_DBUSER_READER

--database-pass-reader <password>
Password for database user having read-only rights. Defaults to reader. May also be controlled by setting environment variable: KUHA_DS_DBPASS_READER

--database-user-editor <user>
Username for database user having editing rights. Defaults to editor. May also be controlled by setting environment variable: KUHA_DS_DBUSER_EDITOR

--database-pass-editor <password>
Password for database user having editing rights. Defaults to `editor`. May also be controlled by setting environment variable: `KUHA_DS_DBPASS_EDITOR`

--loglevel <loglevel>
Lowest logging level of log messages that get output. Valid values are logging levels supported by Python's `logging` `[CRITICAL,ERROR,WARNING,INFO,DEBUG]`. Defaults to `INFO`. May also be controlled by setting environment variable: `KUHA_LOGLEVEL`

--logformat <logformat>
Logging format supported by `logging`. Defaults to `%(asctime)s %(levelname)s(%(name)s): %(message)s)` May also be controlled by setting environment variable: `KUHA_LOGFORMAT`

Configuration file

Args that start with '–' (eg. –document-store-port) can also be set in a config file. The configuration file lookup searches the file from current working directory and from the package directory. The name of the configuration file is `kuha_document_store.ini`.

Note: Invoke with `--help` to print out config file lookup paths.

Environment variables

If the program will be run by using the scripts provided in `scripts` subdirectory, the runtime environment can be controlled via `scripts/runtime_env`, which will be created by copying from `scripts/runtime_env.dist` at installation time by `scripts/install_kuha_document_store_virtualenv.sh`.

3.3.5 Running the program

This guide will use convenience scripts from `scripts` subdirectory. It is assumed that the program was installed by using `scripts/install_kuha_document_store_virtualenv.sh`.

Run Document Store server:

```
./scripts/run_kuha_document_store.sh
```

The script will source `scripts/runtime_env` and activate the installed virtualenv. Finally it calls `kuha_ds_serve`, with given command line arguments.

3.3.6 HTTP API endpoints

Root for the requests is configurable and defaults to `localhost:6001/v0`. Every endpoint will return HTTP status code `500` on internal errors.

Note: Responses with multiple objects will be streamed one object at a time.

REST API

Document Store REST API provides CRUD support to the underlying documents.

GET / (*collection*) **/**
document_id Get object from *collection* with optional *document_id*. If *document_id* is not given, endpoint will return all objects in collection.

Parameters

- **collection** (*str*) – Document collection. One of *studies*, *variables*, *questions* or *study_groups*.
- **document_id** (*str*) – Optional document ID. 24-character hex string.

Status Codes

- 200 OK – Success
- 400 Bad Request – Invalid parameters
- 404 Not Found – Resource not found

POST /studies

Create a new object to studies-collection from JSON request body.

Example request:

```
POST /studies HTTP/1.1
Content-Type: application/json

{"study_number": "study_1"}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type:  application/json; charset=UTF-8

{
    "result": "insert_successful",
    "error": null,
    "affected_resource": "5a82e76e6fb71d06fef00e69"
}
```

Request Headers

- Content-Type – application/json

Request JSON Object

- **study_number** (*string*) – Required study number. Used as an identifier. Must be unique within collection.

Response JSON Object

- **result** (*string*) – Operation outcome.
- **error** (*string*) – Errors during operation.
- **affected_resource** (*string*) – *document_id* of the created object.

Status Codes

- 201 Created – Created successfully.
- 415 Unsupported Media Type – Invalid content type.
- 400 Bad Request – Invalid JSON, Validation failed, Duplicate unique value.

POST /variables

Create a new object to variables-collection from JSON request body.

Example request:

```
POST /variables HTTP/1.1
Content-Type: application/json

{
    "study_number": "study_1",
    "variable_name": "variable_1"
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type:  application/json; charset=UTF-8

{
    "result": "insert_successful",
    "error": null,
    "affected_resource": "5a82ecf16fb71d06fef00e6a"
}
```

Request Headers

- Content-Type – application/json

Request JSON Object

- **study_number** (*string*) – Required study number. Used as an identifier combined with *variable_name*. Their combination must be unique within collection.
- **variable_name** (*string*) – Required variable name. Used as an identifier combined with *study_number*. Their combination must be unique within collection.

Response JSON Object

- **result** (*string*) – Operation outcome.
- **error** (*string*) – Errors during operation.
- **affected_resource** (*string*) – *document_id* of the created object.

Status Codes

- 201 Created – Created successfully.
- 415 Unsupported Media Type – Invalid content type.
- 400 Bad Request – Invalid JSON, Validation failed, Duplicate unique value.

POST /questions

Create a new object to questions-collection from JSON request body.

Example request:

```
POST /questions HTTP/1.1
Content-Type: application/json

{
    "study_number": "study_1",
    "question_identifier": "question_1"
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type:  application/json; charset=UTF-8

{
    "result": "insert_successful",
    "error": null,
    "affected_resource": "5a82ee1a6fb71d06fef00e6b"
}
```

Request Headers

- Content-Type – application/json

Request JSON Object

- **study_number** (*string*) – Required study number. Used as an identifier combined with *question_identifier*. Their combination must be unique within collection.
- **question_identifier** (*string*) – Required variable name. Used as an identifier combined with *study_number*. Their combination must be unique within collection.

Response JSON Object

- **result** (*string*) – Operation outcome.
- **error** (*string*) – Errors during operation.
- **affected_resource** (*string*) – *document_id* of the created object.

Status Codes

- 201 Created – Created successfully.
- 415 Unsupported Media Type – Invalid content type.
- 400 Bad Request – Invalid JSON, Validation failed, Duplicate unique value.

POST /study_groups

Create a new object to study_groups-collection from JSON request body.

Example request:

```
POST /study_groups HTTP/1.1
Content-Type: application/json

{
    "study_group_identifier": "study_group_1"
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type:  application/json; charset=UTF-8

{
    "result": "insert_successful",
    "error": null,
    "affected_resource": "5a82ee876fb71d06fef00e6c"
}
```

Request Headers

- Content-Type – application/json

Request JSON Object

- **study_group_identifier** (*string*) – Required. Used as an identifier and must be unique within collection.

Response JSON Object

- **result** (*string*) – Operation outcome.
- **error** (*string*) – Errors during operation.
- **affected_resource** (*string*) – *document_id* of the created object.

Status Codes

- 201 Created – Created successfully.
- 415 Unsupported Media Type – Invalid content type.
- 400 Bad Request – Invalid JSON, Validation failed, Duplicate unique value.

DELETE / (*collection*) **/**

document_id Delete document or all documents within collection. If optional *document_id* is left out, will delete all documents within collection.

Response JSON Object

- **result** (*string*) – Operation outcome.
- **error** (*string*) – Errors during operation.
- **affected_resource** (*string*) – *document_id* of the created object or number of deleted documents if *document_id* is not given in request.

Status Codes

- 200 OK – Delete successful.
- 404 Not Found – Resource not found.

PUT / (*collection*) **/**

document_id Replace document within collection. :see: *Documents* for information on payload.

Note Leave `_metadata` field out of the request, to let document store handle updated-timestamp automatically.

Request Headers

- Content-Type – application/json

Response JSON Object

- **result** (*string*) – Operation outcome.
- **error** (*string*) – Errors during operation.
- **affected_resource** (*string*) – *document_id* of the created object or number of deleted documents if *document_id* is not given in request.

Status Codes

- 200 OK – Replace successful
- 404 Not Found – Resource not found
- 400 Bad Request – Invalid JSON, Validation failed, Duplicate unique value.

Import API

Documents may be imported to Document Store by using the import API. See *Importers* for more information on how documents are parsed.

POST /import/(*importer_id*)**/**
collection Import document using importer specified with *importer_id*. Optional *collection* may be given to limit the import to a specific collection. If *collection* is not given the importer will import to collections that are applicable to the posted file.

Note Importer API may only be used to initially import documents to the database. After the initial import the documents may be updated via the REST API.

Parameters

- **importer_id** (*str*) – Mandatory parameter to select the importer.
  - ddi_31 imports DDI 3.1 file.
  - ddi_c imports DDI 2.5 file.
  - ddi_122_nesstar imports DDI 1.2.2. Nesstar file.
- **collection** (*str*) – Optional parameter to limit the import to specific collection. One of *studies*, *variables*, *questions* or *study_groups*.

Request Headers

- Content-Type – text/xml

Request body DDI file contents

Response JSON Object

- **result** (*string*) – Operation result
- **imported_docs** (*array*) – Imported document IDs
- **error** (*string*) – Errors found during import

Status Codes

- 200 OK – Import successful
- 400 Bad Request – Empty or invalid request body
- 404 Not Found – Invalid importer id
- 415 Unsupported Media Type – Invalid content type

Query API

Query documents or information on documents from collection.

POST /query/(*collection*)
Execute query against *collection* and return results in JSON.

Request Headers

- Content-Type – application/json

Query Parameters

- **query_type** (*string*) – Optional query parameter to select the query type.
  - select is the default query type. It returns all documents found by filter.

- `count` returns the number of documents which match the filter.
- `distinct` return distinct results for certain field which match the filter.

Request JSON Object

- **`_filter`** (*object*) – Query filter. Used for all query types. Request may specify multiple filter conditions.

Example request with multiple filter conditions:

```
POST /query/variables HTTP/1.1
Content-Type: application/json

{
    "_filter": {
        "study_number": "study_1",
        "variable_name": "variable_1"
    }
}
```

Request JSON Object

- **`fields`** (*array*) – Optional. Select returned fields. Used in select query type. `_id` will always be returned. If not set, full document will be returned.
- **`skip`** (*int*) – Optinal. Skip documents from the beginning. Used in select query type.
- **`limit`** (*int*) – Optional. Limit the number of returned documents. Used in select query type.
- **`sort_by`** (*string*) – Optional. Sort the queried documents by field. Used in select query type.
- **`sort_order`** (*int*) – Optional. Sort order of the queried documents. Used in select query type.
  - `1`: Ascending sort order.
  - `-1`: Descending sort order.
- **`fieldname`** (*string*) – Mandatory for distinct query type. Return distinct values for this field.

Result depends on the requested *query_type*.

JSON response for select query-type

Results will be streamed one object at a time. The object is a document with requested *fields*.

JSON response for count query-type

Response JSON Object

- **`count`** (*int*) – Number of documents found with `_filter`.

JSON response for distinct query-type

If `fieldname` points to document's leaf node the response is in the following format.

```
HTTP/1.1 200 OK
Content-Type:  application/json; charset=UTF-8

{
```

```
    "<fieldname>": ["<list-of-distinct-values>"]
}
```

If `fieldname` points to document's branch node the response is in the following format.

```
HTTP/1.1 200 OK
Content-Type:  application/json; charset=UTF-8

{
    "<fieldname>": ["<list-of-distinct-objects>"]
}
```

Example requests and responses for distinct query-type

```
POST /query/studies?query_type=distinct HTTP/1.1
Content-Type: application/json

{
    "fieldname": "_metadata.updated"
}
```

```
HTTP/1.1 200 OK
Content-Type:  application/json; charset=UTF-8

{
    "_metadata.updated": [
        "2018-02-13T13:49:37Z",
        "2018-02-08T10:55:41Z"
    ]
}
```

```
POST /query/studies?query_type=distinct HTTP/1.1
Content-Type: application/json

{
    "fieldname": "_metadata"
}
```

```
HTTP/1.1 200 OK
Content-Type:  application/json; charset=UTF-8

{
    "_metadata": [
        {
            "updated": "2017-11-09T12:07:48Z",
            "cmm_type": "study",
            "created": "2017-11-09T11:06:03Z"
        },
        {
            "updated": "2017-11-09T11:37:16Z",
            "cmm_type": "study",
            "created": "2017-11-09T11:37:16Z"
        }
    ]
}
```

Note: Distinct queries for datetime-fields will not work as expected, due to different precision in MongoDB and Document Store JSON. MongoDB stores datetimes in millisecond's precision, while Document Store JSON supports second's precision.

Status Codes

- 200 OK – OK
- 400 Bad Request – Message body empty, invalid *query_type*, invalid *query parameters* for query type.
- 415 Unsupported Media Type – Invalid Content-Type

3.3.7 Documents

Documents are objects stored in a collection. Documents support four different types of fields:

1. key-value pair:

```
{"study_number": "1200"}
```

2. contained key-value pairs:

```
{"_metadata": {
    "updated": "2018-01-31T11:37:34Z",
    "cmm_type": "study",
    "created": "2018-01-31T11:37:27Z"
    }
}
```

3. localized contained key-value pairs:

```
{"study_titles": [
   {
       "language": "en",
       "study_title": "Study 1983"
   },
   {
       "language": "fi",
       "study_title": "Tutkimus 1983"
   }
]}
```

4. list of unique values:

```
{"study_numbers": ["1210", "3134", "1175", "2290", "2498"]}
```

3.3.8 Importers

There are importers for DDI3.1., DDI 2.5. and DDI 1.2.2., which can be used to initially import DDI-XML files to document store.

Importer tries to update documents if the are already found from the database. However it is not quaranteed to work properly in cases where an ID element for a field is not found from the DDI. Therefore it is best to use the importer only for initial import of records and afterwards use the REST API to update the documents.

Importer reads xml:lang attributes from the XML-elements to get the language of the element's content. If an element should have no xml:lang attribute, the language is read from the root XML-element's xml:lang. If the root element has no xml:lang attribute the content is assumed to be in english, and `en` is used for the language.

3.4 Kuha OAI-PMH Repo Handler

Kuha OAI-PMH Repo Handler is a HTTP API written in Python for serving Kuha Document Store records through OAI-PMH.

Kuha OAI-PMH Repo Handler is a part of Open Source software bundle Kuha2.

3.4.1 Features

OAI-PMH features:

- Selective harvesting with Sets & Datestamps.
- List request sequence with ResumptionTokens.
- OAI-Identifiers.

Supported metadata standards:

- DDI-C 2.5
- EAD3
- OAI-DC

3.4.2 Dependencies & requirements

- Python 3.5 or newer
- **Recommended**: python3-venv 3.5.1 or newer

The software is continuously tested against Python versions 3.5, 3.6, 3.7, 3.8. and 3.9.

Python packages

The following can be obtained from Python package index.

- tornado (License: Apache License 2.0)
- Genshi (License: BSD)

Kuha Common is a library used with Kuha2 software. It can be obtained from https://bitbucket.org/tietoarkisto/kuha_common

- kuha_common (License: EUPL)

3.4.3 License

Kuha OAI-PMH Repo Handler is available under the EUPL. See LICENSE.txt for the full license.

3.4.4 Configuration

The application can be configured with a configuration file, via command line arguments or by environment variables. If a configuration option is specified in more than one place, then command line values override environment variables which override configuration file values which override defaults.

Note: Configuration options for –oai-pmh-base-url and –oai-pmh-admin-email are required.

Some of the configuration options configure the OAI-PMH repository. Refer to OAI-PMH protocol description for more information.

The following configuration options are available:

-h, --help
Show help message and exit.

--print-configuration
Print active configuration and exit.

--port <port>
Port for serving OAI-PMH Repo Handler. Defaults to `6003` May also be controlled by setting environment variable: `KUHA_OPRH_PORT`.

--template-folder <folder>
Absolute path to Genshi templates. Defaults to `<package-installation-dir>/templates`. May also be controlled by setting environment variable: `KUHA_OPRH_TEMPLATES`.

--oai-pmh-repo-name <repo_name>
OAI-PMH repository name. Defauts to `Kuha2 oai-pmh repository`. May also be controlled by setting environment variable: `KUHA_OPRH_OP_REPO_NAME`.

--oai-pmh-base-url <base_url>
OAI-PMH base url. **Required** configuration value. May also be controlled by setting environment variable: `KUHA_OPRH_OP_BASE_URL`.

--oai-pmh-namespace-identifier <namespace_id>
Namespace identifier to use with OAI-Identifiers. Set `None` to disable use of OAI-Identifiers. Defaults to `None`. May also be controlled by setting environment variable: `KUHA_OPRH_OP_NAMESPACE_ID`.

--oai-pmh-protocol-version <version>
OAI-PMH protocol version. Note that currently only 2.0 is supported. Defaults to `2.0`. May also be controlled by setting environment variable: `KUHA_OPRH_OP_PROTO_VERSION`.

--oai-pmh-results-per-list <results_per_list>
Set maximum number of results for each list response. Defaults to `500`. May also be controlled by setting environment variable: `KUHA_OPRH_OP_LIST_SIZE`.

--oai-pmh-admin-email <email>
OAI-PMH administrator email address. **Required** configuration value. Repeat to give multiple addresses. May also be controlled by setting environment variable: `KUHA_OPRH_OP_EMAIL_ADMIN`.

--oai-pmh-api-version <api_version>
Api version for OAI-PMH Repo Handler. This gets prepended to the URL path. Defaults to `v0`. May also be controlled by setting environment variable: `KUHA_OPRH_API_VERSION`.

--document-store-host <host>
Host & scheme of Kuha Document Store. Defaults to `http://localhost`. May also be controlled by setting environment variable: `KUHA_DS_HOST`.

--document-store-port <port>
Port of Kuha document store database. Defaults to 6001. May also be controlled by setting environment variable: KUHA_DS_PORT.

--document-store-api-version <api_version>
Api version for document store. This gets prepended to the URL path. Defaults to v0. May also be controlled by setting environment variable: KUHA_DS_API_VERSION.

--document-store-client-request-timeout <timeout>
Request timeout (in seconds) for Document Store client. Defaults to 120. May also be controlled by setting environment variable: KUHA_DOCUMENT_STORE_CLIENT_REQUEST_TIMEOUT.

--document-store-client-connect-timeout <timout>
Connect timeout (in seconds) for Document Store client. Defaults to 120. May also be controlled by setting environment variable: KUHA_DOCUMENT_STORE_CLIENT_CONNECT_TIMEOUT.

--document-store-client-max-clients <max_clients>
Maximum number of simultaneous client connections for Document Store client. Defaults to 10. May also be controlled by setting environment variable: KUHA_DOCUMENT_STORE_CLIENT_MAX_CLIENTS.

--loglevel <loglevel>
Lowest logging level of log messages that get output. Valid values are logging levels supported by Python's logging [CRITICAL,ERROR,WARNING,INFO,DEBUG]. Defaults to INFO. May also be controlled by setting environment variable: KUHA_LOGLEVEL

--logformat <logformat>
Logging format supported by logging. Defaults to %(asctime)s %(levelname)s(%(name)s): %(message)s) May also be controlled by setting environment variable: KUHA_LOGFORMAT

Configuration file

Args that start with '–' (eg. –document-store-port) can also be set in a config file. The configuration file lookup searches the file from current working directory and from the package directory. The name of the configuration file is kuha_oai_pmh_repo_handler.ini.

Note: Invoke with --help to print out config file lookup paths.

Environment variables

If the program will be run by using the scripts provided in scripts subdirectory, the runtime environment can be controlled via scripts/runtime_env, which will be created by copying from scripts/runtime_env.dist at installation time by scripts/install_kuha_oai_pmh_repo_handler_virtualenv.sh.

3.4.5 Running the Server

This guide will use convenience scripts from scripts subdirectory. It is assumed that the program was installed by using scripts/install_kuha_oai_pmh_repo_handler_virtualenv.sh.

Run OAI-PMH Repo Handler server:

```
./scripts/run_kuha_oai_pmh_repo_handler.sh --oai-pmh-base-url=<base-url> --oai-pmh-
↪admin-email=<admin-email>
```

The script will source scripts/runtime_env and activate the installed virtualenv. Finally it calls kuha_oai_serve, with given command line arguments.

3.4.6 Ensuring OAI-PMH serves correct records

The program contains a helper script to run through all records from OAI-PMH Repo Handler using OAI verb ListRecords. The script will print out all identifiers it encounters and log out the time it took to complete the full ListRecords sequence. Note that the OAI-PMH Repo Handler server must be running and accessible in order to get correct results.

If any error conditions are encountered the best place to determine the cause is Kuha OAI-PMH Repo Handler server log.

Run through all records using `oai_dc` metadataprefix:

```
./scripts/list_records.sh oai_dc
```

See help for more information and configuration options:

```
./scripts/list_records.sh --help
```

3.5 Kuha OSMH Repo Handler

Kuha OSMH Repo Handler is a HTTP API written in Python for serving Kuha Document Store records through OSMH.

Kuha OSMH Repo Handler is a part of Open Source software bundle Kuha2.

3.5.1 Features

OSMH record types supported:

- Study
- Variable
- Question
- StudyGroup

OSMH Repo Handler supports streaming responses. Note that the requesting party must also support streaming. Streaming is disabled by default.

3.5.2 Dependencies & requirements

- Python 3.5 or newer
- **Recommended**: python3-venv 3.5.1 or newer

The software is continuously tested against Python versions 3.5, 3.6, 3.7, 3.8. and 3.9.

Python packages

The following can be obtained from Python package index.

- tornado (License: Apache License 2.0)

Kuha Common is a library used with Kuha2 software. It can be obtained from https://bitbucket.org/tietoarkisto/kuha_common

- kuha_common (License: EUPL)

3.5.3 License

Kuha OSMH Repo Handler is available under the EUPL. See LICENSE.txt for the full license.

3.5.4 Configuration

The application can be configured with a configuration file, via command line arguments or by environment variables. All configuration options have default values. If a configuration option is specified in more than one place, then command line values override environment variables which override configuration file values which override defaults.

The following configuration options are available:

-h, --help
Show help message and exit.

--print-configuration
Print active configuration and exit.

--stream-response
Switch to enable streaming from ListRecordHeaders endpoint. Defaults to `False`. May also be controlled by setting environment variable: `KUHA_OSMH_STREAM_RESPONSE`.

--port `<port>`
Port for serving OSMH Repo Handler. Defaults to `6002` May also be controlled by setting environment variable: `KUHA_ORH_PORT`.

--osmh-repo-handler-api-version `<api_version>`
Api version for OSMH Repo Handler. This gets prepended to the URL path. Defaults to `v0`. May also be controlled by setting environment variable: `KUHA_OSMH_API_VERSION`.

--query-limit `<limit>`
Optional query limit for configuring the limit of records per query when fetching multiple records from Document Store. Set 0 to disable. Defaults to `0`. May also be controlled by setting environment variable: `KUHA_OSMH_QUERY_LIMIT`.

--document-store-host `<host>`
Host & scheme of Kuha Document Store. Defaults to `http://localhost`. May also be controlled by setting environment variable: `KUHA_DS_HOST`.

--document-store-port `<port>`
Port of Kuha document store database. Defaults to `6001`. May also be controlled by setting environment variable: `KUHA_DS_PORT`.

--document-store-api-version `<api_version>`
Api version for document store. This gets prepended to the URL path. Defaults to `v0`. May also be controlled by setting environment variable: `KUHA_DS_API_VERSION`.

--document-store-client-request-timeout `<timeout>`
Request timeout (in seconds) for Document Store client. Defaults to `120`. May also be controlled by setting environment variable: `KUHA_DOCUMENT_STORE_CLIENT_REQUEST_TIMEOUT`.

--document-store-client-connect-timeout `<timout>`
Connect timeout (in seconds) for Document Store client. Defaults to `120`. May also be controlled by setting environment variable: `KUHA_DOCUMENT_STORE_CLIENT_CONNECT_TIMEOUT`.

--document-store-client-max-clients `<max_clients>`
Maximum number of simultaneous client connections for Document Store client. Defaults to `10`. May also be controlled by setting environment variable: `KUHA_DOCUMENT_STORE_CLIENT_MAX_CLIENTS`.

--loglevel <loglevel>
Lowest logging level of log messages that get output. Valid values are logging levels supported by Python's logging [CRITICAL,ERROR,WARNING,INFO,DEBUG]. Defaults to INFO. May also be controlled by setting environment variable: KUHA_LOGLEVEL

--logformat <logformat>
Logging format supported by logging. Defaults to %(asctime)s %(levelname)s(%(name)s): %(message)s) May also be controlled by setting environment variable: KUHA_LOGFORMAT

Configuration file

Args that start with '–' (eg. –document-store-port) can also be set in a config file. The configuration file lookup searches the file from current working directory and from the package directory. The name of the configuration file is kuha_osmh_repo_handler.ini.

Note: Invoke with --help to print out config file lookup paths.

Environment variables

If the program will be run by using the scripts provided in scripts subdirectory, the runtime environment can be controlled via scripts/runtime_env, which will be created by copying from scripts/runtime_env.dist at installation time by scripts/install_kuha_osmh_repo_handler_virtualenv.sh.

3.5.5 Running the program

This guide will use convenience scripts from scripts subdirectory. It is assumed that the program was installed by using scripts/install_kuha_osmh_repo_handler_virtualenv.sh.

Run OSMH Repo Handler server:

```
./scripts/run_kuha_osmh_repo_handler.sh
```

The script will source scripts/runtime_env and activate the installed virtualenv. Finally it calls kuha_osmh_serve, with given command line arguments.

3.6 Kuha Client

Kuha Client is used to submit records to Kuha Document Store. Kuha Client is written in Python and uses HTTP to communicate with Document Store.

3.6.1 Features

- Support for DDI 3.1, DDI 2.5 and DDI 1.2.2 metadata standards.
- Import records to Document Store.
- Update records already stored in Document Store.
- Delete records in Document Store.
- Batch process DDI files by recursing into directories:
  - Option to remove records from Document Store not found in the current batch.

  - Option to keep track of previously processed files and bypass processing if modification times have not changed.

3.6.2 Dependencies & requirements

- Python 3.5. or newer
- **Recommended:** python3-venv 3.5.1 or newer

The software is continuously tested against Python versions 3.5, 3.6, 3.7, and 3.8.

Python packages

Kuha Common is a library used with Kuha2 software. It can be obtained from https://bitbucket.org/tietoarkisto/kuha_common

- kuha_common (License: EUPL)

3.6.3 License

Kuha Client is available under the EUPL. See LICENSE.txt for the full license.

3.6.4 Configuration

Most common configuration options are described here. Use `--help` to print all available options.

`paths`
: **Required** positional argument. Absolute path to file or directory. Repeat to process multiple paths.

`-h, --help`
: Show help and exit.

`--collection` `<collection>`
: **Only for upsert and import run.** Limits the import to a spesific document type. Valid values are `[studies, variables, questions, study_groups]`. Set `None` to import all document types. Defaults to `None`.

`--document-store-url` `<document_store_url>`
: **Required**. Full URL to Document Store, for example `http://localhost:6001/v0`. May also be controlled by setting environment variable: `KUHA_DS_URL`.

`--file-log-path` `<path>`
: **Only for upsert and import run.** Store processed files to file log. Compare modification times on subsequent run. Bypass if modification times have not changed.

`--remove-absent`
: **Only for upsert run.** Remove records from Document Store not present in current batch.

3.6.5 Running the program

If installed to a Python virtual environment, the environment must be activated before running the program.

Import records to Document Store by scanning a directory tree for .xml files to submit and create a file-log to keep track of processed files:

```
python -m kuha_client.kuha_import --file-log-path=file_log /path/to/directory
```

Upsert records (insert and update) to Document Store by scanning a directory tree for .xml files and comparing found files to the ones store in file-log. If a file's modification time is newer than the one stored in file-log, the file gets processed. When using the –remove-absent flag, any ID found from document store, but not from the current batch, gets removed:

```
python -m kuha_client.kuha_upsert --file-log-path=file_log --remove-absent /path/to/
↪directory
```

Delete record from collection:

```
python -m kuha_client.kuha_delete studies 5af94ff06fb71d7646160bd4
```

Delete all records from collection:

```
python -m kuha_client.kuha_delete studies ALL
```

Delete all records from all collections:

```
python -m kuha_client.kuha_delete ALL ALL
```

3.7 Kuha Common

Kuha Common is a Python library used with Kuha2 software bundle.

3.7.1 Dependencies & requirements

Versions spesified here are the ones that the software has been developed with. Newer versions may be compatible.

- Python 3.5
- **Recommended:** python3-venv 3.5.1

Python packages

The following can be obtained from Python package index.

- ConfigArgParse (License: MIT)
- Tornado (License: Apache License 2.0)

3.7.2 License

Kuha Common is available under the EUPL. See LICENSE.txt for the full license.

3.8 Developer Documentation

3.8.1 kuha_common

High-level modules common for Kuha applications.

server.py

Common server functions & classes used in Kuha.

kuha_common.server.**log_request**(*handler*)
Log request output to JSON. Gets called after each successful response.

Parameters **handler** (subclass of tornado.web.RequestHandler) – handler for the current request.

kuha_common.server.**log_exception**(*typ*, *value*, *tb*, *correlation_id*)
Log exception output to JSON. Gets called from *RequestHandler*.

Parameters

- **typ** – type of exception
- **value** – caught exception
- **tb** – traceback
- **correlation_id** – correlation id of the request that ended in exception.

kuha_common.server.**str_api_endpoint**(*api_version*, *suffix=None*)
Helper function to prepend endpoints with api_version.

Parameters

- **api_version** (*str*) – version of the api.
- **suffix** (*str*) – api endpoint.

Returns str – endpoint prepended with api_version

kuha_common.server.**serve**(*web_app*, *port*, *process_count=0*, *on_exit=None*)
Serve web application.

Parameters

- **web_app** (tornado.web.Application) – application to serve
- **port** (*int*) – Port to listen to.
- **process_count** (*int*) – number of processes to spawn. 0 = forks one process per cpu.
- **on_exit** (*function*) – callback on server/ioloop stop.

class kuha_common.server.**RequestHandler**(**args*, ***kwargs*)
Common request handler for kuha server applications. Subclass in application specific handlers.

prepare()
Prepare each request.

Look for correlation id; create one if not found. Set correlation id to response header.

set_output_content_type(*ct*, *charset='UTF-8'*)
Sets content type for responses.

Parameters

- **ct** (*str*) – content type for response.
- **charset** (*str*) – charset definition for response content type.

log_exception(*typ*, *value*, *tb*)
Overrides tornados exception logging. Sends HTTP errors as responses. Calls customised *log_exception()* which outputs JSON encoded log messages with request correlation ids. For easier

debugging it also calls `tornado.web.RequestHandler.log_exception` to output full traceback.

Parameters

- **typ** – type of exception
- **value** – caught exception
- **tb** – traceback

assert_request_content_type(*supported_content_type*)
Assert request has correct content type header.

Parameters **supported_content_type** (*str*) – content type supported by endpoint.

Raises ***InvalidContentType*** – if request has invalid content type.

write_error(*status_code*, ***kwargs*)
Overrides `tornado.web.RequestHandler.write_error`. Outputs error messages in preferred content type.

Parameters

- **status_code** (*int*) – HTTP status code.
- ****kwargs** – keyword arguments are passed to `tornado.web.RequestHandler.write_error` if output content type is not application/json

class kuha_common.server.**WebApplication**(*handlers*)
Override `tornado.web.Application` to make sure server applications are using correct initialization parameters.

log_request(*handler*)
Override `tornado.web.Application.log_request`. Server uses it's own implementation of log_request.

Parameters **handler** (Subclass of `tornado.web.RequestHandler`) – Handler of current request.

exception kuha_common.server.**KuhaServerError**(*msg*, *status_code=500*, *context=None*)
Base class for common HTTP-exceptions

exception kuha_common.server.**InvalidContentType**(*requested_content_type*, *supported_content_type*)
Invalid content type HTTP-exception.

exception kuha_common.server.**BadRequest**(*msg=None*)
Bad request HTTP-exception.

exception kuha_common.server.**ResourceNotFound**(*msg=None*, *context=None*)
Resource not found HTTP-exception.

query.py

Perform query operations against Kuha Document Store.

Offers High-level query methods to facilitate an easy access point with all necessary actions and properties needed to perform queries. To query the Document Store the caller only needs to use methods defined in class QueryController and records defined in *kuha_common.document_store.records*

class kuha_common.query.**ResultHandler**(*record_constructor*, *on_record=None*)
Class which handles results and correct calls to callbacks, if any. Stores the result for later use.

Dynamically creates callable method `handle()` which receives result payload and calls `on_record` correctly.

Parameters

- **record_constructor** (*kuha_common.document_store.records.Study* or *kuha_common.document_store.records.Variable* or *kuha_common.document_store.records.Question* or *kuha_common.document_store.records.StudyGroup*) – Class to construct Document Store record.
- **on_record** (*function or coroutinefunction or None*) – Callback called with constructed record instance as parameter.

Returns *ResultHandler*

class kuha_common.query.**QueryController**(*headers=None*, *record_limit=0*)

Asynchronous controller to query the Document Store.

Use to build queries and automatically fetch responses using HTTP as a protocol and JSON as exchange format.

Optional record_limit parameter may be given at initialization time to limit the number of records that are requested in a single HTTP request.

Parameters

- **headers** (*dict*) – Optional headers parameter to store headers that are used in each query application wide as HTTP headers.
- **record_limit** (*int*) – Optional record_limit parameter which is used to limit the number of records requested in a single HTTP request.

Example:

```
from kuha_common.document_store import Study
query_ctrl = QueryController()
study = await query_ctrl.query_single(
    Study,
    fields=[Study._metadata, Study.study_number],
    _filter={Study.study_number: 1234}
)
```

fk_constants

alias of `FilterKeyConstants`

query_single(*record*, *on_record=None*, *headers=None*, ***kwargs*)

Query single record.

Parameters

- **record** (Subclass of *kuha_common.document_store.records.RecordBase*) – class used to construct the record instance.
- **on_record** (*function or coroutinefunction*) – Optional callback function that gets passed the returned and instantiated record object.
- **headers** (*dict*) – Optional headers for this query. Headers get added to headers given for QueryController at initialization time. Note that it will overwrite headers with same key.
- ****kwargs** – Keyword arguments contain parameters for the query. They get passed to *kuha_common.document_store.query.Query.construct()*

Returns None if passed on_record callback, else returns the initiated record object.

Raises ***QueryException*** – if query parameters given as keyword arguments contain limit-parameter.

query_multiple(*record*, *on_record*, *headers=None*, ***kwargs*)
Query multiple records.

Queries the document store for multiple records. Behaviour depends on whether `record_limit` has been set:

If there is a record_limit

- the queries are split in multiple HTTP requests and queued.
- This method returns a *kuha_common.document_store.client.JSONStreamClient.run_queued_requests()*, which is to be called without arguments when the queries are to be executed.
- on_record must be a normal function that takes the constructed record instance as parameter.

If there is no record_limit

- this method returns nothing.
- The `on_record` callback gets called with each instantiated record object.
- `on_record` may be a normal function or a coroutine.

Parameters

- **record** (Subclass of *kuha_common.document_store.records.RecordBase*) – class used to construct the record instance.
- **on_record** (*function or coroutine*) – callback that gets called with each instantiated record instance.
- **headers** (*dict*) – optional headers used for this query.
- ****kwargs** – Keyword arguments contain parameters for the query. They get passed to *kuha_common.document_store.query.Query.construct()*

Returns None if no `record_limit`, else *kuha_common.document_store.client.JSONStreamClient.run_queued_requests()*

query_count(*record*, *headers=None*, ***kwargs*)
Query the number of records.

Parameters

- **record** (Subclass of *kuha_common.document_store.records.RecordBase*) – class used to construct the record instance.
- **headers** (*dict*) – optional headers used for this query.
- ****kwargs** – Keyword arguments contain parameters for the query. They get passed to *kuha_common.document_store.query.Query.construct()*

Returns Number of records

Return type int

query_distinct(*record*, *headers=None*, ***kwargs*)
Query distinct values.

Parameters

- **record** (Subclass of *kuha_common.document_store.records.RecordBase*) – record to query for.
- **headers** (*dict*) – optional headers used for this query.
- ****kwargs** – Keyword arguments contain parameters for the query. They get passed to *kuha_common.document_store.query.Query.construct_distinct()*

Returns distinct values: {fieldname : [value1, value2, value3]}. Note that contained values may be dictionaries or strings, depending on what is stored in requested field.

Return type dict

cli_setup.py

Command line setup for Kuha applications

Parse command line for common configuration options and store loaded settings.

Load modules for querying Document Store:

```
import os
from kuha_common import cli_setup
cli_setup.load(os.getcwd())
settings = cli_setup.setup(cli_setup.MOD_DS_CLIENT, cli_setup.MOD_DS_QUERY)
```

kuha_common.cli_setup.**MOD_DS_CLIENT = 'document_store.client'**
Constant for configuring *kuha_common.document_store.client*

kuha_common.cli_setup.**MOD_DS_QUERY = 'document_store.query'**
Constant for configuring *kuha_common.document_store.query*

kuha_common.cli_setup.**MOD_LOGGING = 'logging'**
Constant for configuring logging

kuha_common.cli_setup.**add_kuha_loglevel**(*parser=None*)
Add loglevel to parser

Parameters **parser** (*ArgumentParser instance*) – command line parser.

kuha_common.cli_setup.**add_kuha_logformat**(*parser=None*)
Add logformat to parser

Parameters **parser** (*ArgumentParser instance*) – command line parser.

kuha_common.cli_setup.**add_document_store_url**(*parser=None*, ***kw*)
Add document store url to parser

Parameters

- **parser** (*ArgumentParser instance*) – command line parser.
- ****kw** – keyword arguments get passes to parser.add

kuha_common.cli_setup.**add_document_store_host**(*parser=None*)
Add document store host to parser

Parameters **parser** (*ArgumentParser instance*) – command line parser.

kuha_common.cli_setup.**add_document_store_port**(*parser=None*)
Add document store port to parser

Parameters **parser** (*ArgumentParser instance*) – command line parser.

kuha_common.cli_setup.**add_document_store_api_version**(*parser=None*)
Add document store api-version to parser

Parameters **parser** (*ArgumentParser instance*) – command line parser.

kuha_common.cli_setup.**add_document_store_client_request_timeout**(*parser=None*)
Add document store client request timeout to parser

Parameters **parser** (*ArgumentParser instance*) – command line parser.

kuha_common.cli_setup.**add_document_store_client_connect_timeout**(*parser=None*)
Add document store client connect timeout to parser

Parameters **parser** (*ArgumentParser instance*) – command line parser

kuha_common.cli_setup.**add_document_store_client_max_clients**(*parser=None*)
Add document store client max clients timeout to parser

Parameters **parser** (*ArgumentParser instance*) – command line parser.

kuha_common.cli_setup.**add_print_configuration**(*parser=None*)
Add print configuration helper for testing configuration options.

Parameters **parser** (*ArgumentParser instance*) – command line parser.

kuha_common.cli_setup.**add_server_process_count_configuration**(*parser=None*)
Add tornado server process count to configuration options parser.

Parameters **parser** (*ArgumentParser instance*) – command line parser.

class kuha_common.cli_setup.**KuhaConfigFileParser**
Inherit to override `configargparse.DefaultConfigFileParser.get_syntax_description()`

get_syntax_description()
Override syntax description of `configargparse.DefaultConfigFileParser`

Returns Syntax description for configuration file.

Return type str

class kuha_common.cli_setup.**Settings**
Class for command line settings.

is_parser_loaded()
Check is parser loaded.

Return type bool

is_settings_loaded()
Check is settings loaded.

Return type bool

set_abs_dir_path(*path*)
Set absolute directory path of configurable kuha application.

Parameters **path** (*str*) – absolute path to kuha application directory.

get_abs_dir_path()
Return absolute directory path of configurable kuha application.

Returns absolute path to directory.

Return type str

add_logging_configs()
Wrapper to add logging-module configuration.

add_document_store_query_configs()
Wrapper to add document_store_query configuration.

add_document_store_client_configs()
Wrapper to add document_store_client configuration.

setup_logging()
Setup `logging` module.

setup_document_store_query()
Setup *kuha_common.document_store.query* module.

setup_document_store_client()
Setup *kuha_common.document_store.client* module.

load_parser(*config_file=None*, ***kw*)
Load command line parser.

Additional keyword arguments are passed to `configargparse.ArgumentParser`.

Parameters **config_file** (*str*) – Name of configuration file.

set(*parsed_opts*)
Assign parser options to settings.

Parameters **parsed_opts** (`argparse.Namespace`) – parser options.

load_cli_args()
Load command line arguments.

get()
Return active settings.

Returns active settings.

Return type `argparse.Namespace`

add(**args*, ***kwargs*)
Add item for parser. Settings must not yet be loaded but parser must be loaded.

Parameters

- ***args** – arguments passed to `configargparse.ArgumentParser`
- ****kwargs** – keyword arguments passed to `configargparse.ArgumentParser`

kuha_common.cli_setup.**setup**(**modules*)
Setup command line parser.

Load modules, parse command line arguments, return loaded settings in `argparse.Namespace`

Parameters ***modules** (*str*) – common Kuha modules to load and include in parsing of command line arguments.

Returns Loaded settings.

Return type `argparse.Namespace`

kuha_common.cli_setup.**get_settings**()
Get loaded settings stored in *Settings*.

Returns Loaded settings.

Return type `argparse.Namespace`

kuha_common.cli_setup.**add**(**args*, ***kwargs*)
Module level function to add items to be parsed in *Settings* singleton.

Parameters

- ***args** – arguments passed to *Settings.add()*.
- ****kwargs** – keyword arguments passed to *Settings.add()*

kuha_common.cli_setup.**load**(*abs_dir_path*, ***kwargs*)
Module level function to load parser to *Settings* singleton.

Parameters

- **abs_dir_path** (*str*) – absolute path to directory of the kuha application to be configured.
- ****kwargs** – keyword arguments passed to *Settings.load_parser()*.

kuha_common.cli_setup.**prepend_abs_dir_path**(*path*)
Helper function to prepend the stored absolute directory path to given argument.

Parameters **path** – end of the path.

Returns absolute path ending to path.

Return type str

document_store

Contains modules for interacting with Document Store.

document_store/client.py

kuha_common.document_store.client provides a http client interface to communicate with Kuha Document Store.

class kuha_common.document_store.client.**JSONStreamClient**
Base class used for requests. Implements own queue to store requests, since tornado.httpclient.AsyncHTTPClient starts timers for request_timeout and connect_timeout at the moment we call client.fetch(). See https://github.com/tornadoweb/tornado/issues/1400 for more details.

Handles JSON decoding of incoming chunks and the encoding of body to JSON.

max_clients = 10
Controls the maximum number of concurrent clients.

request_timeout = 120
Sets timeout for a request.

connect_timeout = 120
Sets timeout for establishing a connection.

sleep_on_queue = 5
Sets sleep timer for queue.

classmethod set_max_clients(*max_clients*)
Set maximum concurrent clients

classmethod set_request_timeout(*request_timeout*)
Set timeout per request

classmethod set_connect_timeout(*connect_timeout*)
Set timeout per connection

classmethod request(*url*, ***kwargs*)
Constucts a streaming request.

Parameters

- **url** (*str*) – url to request.
- **kwargs** – keyword arguments passed to `tornado.httpclient.HTTPRequest`

Returns `tornado.httpclient.HTTPRequest`

wrap_streaming_callback(*callback*)
Wrap streaming callback to support chunked JSON responses.

Parameters **callback** (*callable.*) – streaming callback. Gets called with response which is decoded to python object from JSON.

Returns Wrapped callback

Return type `functools.partial`

execute_stored_callbacks()
Executes asynchronous callbacks stored in `_callbacks`

run_queued_requests(*queued_requests=None*)
Run queued requests.

Calls queued requests asynchronically. Sleeps for *sleep_on_queue* if *max_clients* reached.

Parameters **queued_requests** (`collections.deque`) – Optionally pass queued_requests to run.

get_streaming_request(*streaming_callback*, *url*, *body=None*, *method=None*, *headers=None*, ***kw*)
Get a streaming request.

Sets default headers `Content-Type: application/json` if not already given. Encodes body to JSON if given and is not string or bytes. If response is empty (for example query with no results) the streaming callback doesn't get called.

Subclass and override to support arbitrary requests.

Parameters

- **streaming_callback** (*callable*) – callback which receives the response if any.
- **url** (*str*) – URL to send request to.
- **body** (*str, dict, list, tuple, integer, float or None*) – Optional request body. String will be supplied as is. Other values will be encoded to JSON.
- **method** (*str or None*) – HTTP method. Defaults to `POST`.
- **headers** (*dict or None*) – optional request headers. if Content-Type is not set, will set 'Content-Type': 'application/json' as default.

Returns HTTP request

Return type `tornado.httpclient.HTTPRequest`

queue_request(**args*, ***kwargs*)
Queue request to be run aynchronously by calling `run_queued_requests`.

Parameters

- ***args** – arguments passed to `get_streaming_request`
- ****kwargs** – keyword arguments passed to `get_streaming_request`.

Returns *run_queued_requests()* method to call to run the queued requests.

fetch(**args*, ***kwargs*)
Run single query.

Parameters

- ***args** – arguments passed to `queue_requests`.
- ****kwargs** – keyword arguments passed to `queue_requests`

document_store/query.py

Access query properties by convenience methods to help build valid queries against the Document Store.

class kuha_common.document_store.query.**FilterKeyConstants**
Class used as a namespace to contain constants used in query filter.

exception kuha_common.document_store.query.**QueryException**(*msg*, *context=None*)
Exception for errors raised by *Query*. Has an optional context parameter for giving some additional information about the context of the exception.

class kuha_common.document_store.query.**Query**(*query*, *query_document*, *query_type='select'*)
Manipulate query properties without compromising the validity of the constructed query. Build the correct url for different query types.

Note This class provides low-level operations. Use *kuha_common.query.QueryController* for easy access to query actions and properties.

Example:

```
from kuha_common.document_store import Study, Query
query = Query(Query.construct(_filter={Study.study_number:'123'}), Study.
↪collection)
```

Parameters

- **query** (*dict*) – Actual query containing the properties such as _filter, fields, sort_by etc.
- **query_document** (*str*) – One of the supported query documents declared in `Query.supported_query_documents` and specified in `kuha_common.document_store.records.py`
- **query_type** (*str*) – Optional query_type parameter. Defaults to *Query.query_type_select*. Other valid values are *Query.query_type_count* and *Query.query_type_distinct*.

k_filter = '_filter'
Query parameter for filtering results.

k_fields = 'fields'
Query parameter for fields to contain in results.

k_limit = 'limit'
Query parameters for limiting returned results.

k_skip = 'skip'
Query parameter for skipping number of results from the beginning of the resultset.

k_sort_order = 'sort_order'
Query parameter for sort order.

k_sort_by = 'sort_by'
Query parameter for sorting by a certain field.

k_fieldname = 'fieldname'
Query parameter for distinct queries. Specifies a field from which the distinct values are to be fetched.

query_type_select = 'select'
Query type for *select* queries. Using this query type gets records as a response.

query_type_count = 'count'
Query type for count queries. Using this query type the query returns an integer.

query_type_distinct = 'distinct'
Query type for distinct queries. Using this query type the returning object contains all distinct values for a certain field.

classmethod set_base_url(*base_url*)
Configure document store url used as a base when constructing the endpoint url for queries.

classmethod as_supported_datetime_str(*datetime_obj*)
Get datetime object as a supported datetime string.

Parameters **datetime_obj** (*datetime-object.*) – Python datetime-object to be convered to str.

Returns String represenation of the datetime-object that is suitable foe querying.

Return type str

classmethod construct(***kwargs*)
Construct valid query parameters.

Example:

```
from kuha_common.document_store import Query, Study
params = Query.construct(_filter={Study.study_number:'123'},
                         fields=[Study._metadata, Study._id, Study.abstract],
                         sort_by=Study._id)
query = Query(params, Study.collection)
```

Parameters ****kwargs** – keys should be valid query properties, while values should hold corresponding query values supported by the key.

Returns Valid query, ready to be sent to Document Store.

Return type dict

classmethod construct_distinct(***kwargs*)
Construct valid query parameters for distinct queries.

Parameters ****kwargs** – keys should be valid query properties, while values should hold corresponding query values supported by the key.

Returns Valid query, ready to be sent to Document Store.

Return type dict

classmethod build_query_for_date_range(*from_=None*, *until=None*)
Build query filter for date-range.

Parameters

- **from** (*datetime-object*) – start of the date-range:
- **until** (*datetime-object*) – end of the date-range:

Returns date-range query-filter with datetime-objects converted into string representation.

Return type dict

classmethod build_query_for_exists(*exists*)
Build query for exists-query.

Parameters **exists** (*bool*) – whether the field should exists or not.

Returns valid exists query for filter.

Return type dict

Raises ValueError for invalid boolean values in exists-parameter.

classmethod get_valid_params(*query_type=None*)
Return valid query parameters for the query type.

Parameters **query_type** (*str*) – Optional query_type for which the query-parameters should be valid for.

classmethod is_valid_query(*query*, *query_type*)
Check the validity of query parameters.

Parameters

- **query** (*dict*) – Full query to validate.
- **query_type** (*str*) – Query type to validate against.

Returns Whether or not the query-parameters given are valid.

Return type bool

classmethod is_valid_query_type(*query_type*)
Check the validity of query_type.

Parameters **query_type** (*str*) – Query type to validate.

Returns Whether or not the query type is valid.

Return type bool

is_valid_query_document(*query_document*)
Check the validity of query document.

Parameters **query_document** (*str*) – Query document to validate.

Returns Whether or not the query document is valid.

Return type bool

is_valid_param(*parameter*)
Check the validity of a single query parameter.

Parameters **parameter** (*str*) – Query parameter to validate.

Returns Whether or not the parameter is valid.

Return type bool

validate_query(*query*)
Validate query parameters.

Checks parameters' validity for chosen query type. Raises *QueryException* if invalid.

Parameters **query** (*dict*) – Query parameters.

Returns Query parameters.

Return type dict

Raises *QueryException* if query parameters are invalid.

validate_query_type(*query_type*)
Validate query type.

Checks that the query type is supported by Document Store. Raises *QueryException* for invalid query type.

Parameters **query_type** (*str*) – Query type to validate.

Returns Query type.

Return type str

Raises *QueryException* if query type is invalid.

validate_query_document(*query_document*)
Validates query document.

Checks that the query document is supported by Document Store. Raises *QueryException* if invalid.

Parameters **query_document** (*str*) – Query document to validate.

Returns Query document

Return type str

Raises *QueryException* if query document is invalid.

get_endpoint()
Get correct endpoint for querying the Document Store.

Builds the endpoint by consulting configured values and the instantiated query for query_type and query_document

Returns Full url to Document Store endpoint which handles the constructed query.

Return type str

get_query(*strip_invalid_params=True*)
Returns the constructed query parameters.

If the query type has been changed after initialization, for example to get the count of records, this method strips the invalid query parameters from the returned query. When doing so, it does not change the stored query parameters, but rather makes a copy of them for manipulating and returning.

Parameters **strip_invalid_params** (*bool*) – Whether to strip the unsupported (=invalid) query parameters out of the returned query.

Returns Constructed query parameters ready to submit to Document Store.

Return type dict

get_limit()
Get query limit parameter.

Returns Query limit (int) if set. None if not set.

Return type int or None

get_skip()
Get query skip parameter.

Returns Query skip (int) if set. None if not set.

Return type int or None

set_limit(*limit*)
Set limit parameter for query.

Limit controls how many results should be returned.

Parameters **limit** (*int*) – Limit parameter for query.

Returns self for easy aggregation of manipulation methods.

Return type instantiated *Query()*

set_skip(*skip*)
Set skip parameter for query.

Skip conrols how many results should be skipped from the start (offset).

Parameters **skip** (*int*) – Skip parameter for query.

Returns self for easy aggregation of manipulation methods.

Return type instantiated *Query()*

set_fields(*fields*)
Set fields parameter for query.

Field controls which fields of the record should be returned. *fields* can be a list of strings in the form used by MongoDB or a list of *kuha_common.document_store.records* class-variables.

Example:

```
from kuha_common.document_store import Query, Study
_params = Query.construct(_filter={Study.study_number:'123'})
_query = Query(_params, Study.collection)
_query.set_fields([Study.abstract, Study.study_number])
```

Parameters **fields** (*list*) – Fields parameter for query.

Returns self for easy aggregation of manipulation methods.

Return type instantiated *Query()*

set_sort_by(*sort_by*)
Set sort_by parameter for query.

Determines sorting of the returned results. *sort_by* can be a string in the form used by MongoDB or a *kuha_common.document_store.records* class-variables.

Parameters **sort_by** (*srt or class-variable of a record.*) – Sort by parameter for query.

Returns self for easy aggregation of manipulation methods.

Return type instantiated *Query()*

set_sort_order(*order*)
Set sort order for the query.

Determines the order which the returned results are to be sorted by.

Note Valid values come from pymongo. They actually depend on the mongodb driver, but since this is a caller API we don't want to make pymongo a dependency.

Parameters **order** (*int*) – Sort order. Must be either 1 or -1.

Returns self for easy aggregation of manipulation methods.

Return type instantiated *Query()*

Raises *QueryException* for invalid order values.

set_query_type(*query_type*)
Set query type.

Parameters **query_type** (*str*) – Valid query type for the query to be constructed.

Returns self for easy aggregation of manipulation methods.

Return type instantiated *Query()*

add_query_statement(*field*, *statement*)
Add query statement.

Manipulates the _filter parameter of the query parameters. Raises a *QueryException* if the field already has a statement declared in _filter.

Parameters

- **field** (*str*) – Field to target the statement to.
- **statement** (*str*) – Statement to filter the results by.

Returns self for easy aggregation of manipulation methods.

Return type instantiated *Query()*

add_query_statements(***kwargs*)
Add multiple query statements to filter the returned results.

Manipulates the _filter parameter of the query parameters.

Parameters ****kwargs** – key-value pairs that are to be added to the _filter parameter.

Returns self for easy aggregation of manipulation methods.

Return type instantiated *Query()*

document_store/field_types.py

Properties and actions for field types supported by records defined in *kuha_common.document_store.records*

Provides field types to be used not only for the construction of new records and updating exising records, but also to provide a format for fields of records that is interchangeable in a way that a receiver does not need to know the specifics of a field beforehand, but may use the field to gain knowledge of the properties of the field.

This module also provides factories which are used to fabricate the fieldtypes. The instantiated factories hold knowledge of the fields even thought the fields themselves are not yet instantiated. This knowledge is used for querying records, but also to dynamically fabricate fieldtypes for records in *kuha_common.document_store.records*

exception kuha_common.document_store.field_types.**FieldTypeException**
Exception to raise on field type errors. Used for programming errors.

class kuha_common.document_store.field_types.**Value**(*name*, *value=None*)
Value is the most simple type of field.

Field type with name and single value. Serves also as a baseclass for other field types.

Parameters

- **name** (*str*) – Name of the field.
- **value** – Optional value for the field.

set_value(*value*)
Set initial value.

Parameters **value** – Value to set.

add_value(*value*)
Add value for the field.

Note Overrides existing value.

Parameters **value** – The value to set.

get_value()
Get the value of the field.

Returns Value.

get_name()
Get name of the field.

Returns The name of the field.

Return type str

export_dict()
Exports Value as dictinary.

Returns {name:value}

Return type dict

import_records(*record*)
Import record to field.

Parameters **record** – record to import.

updates(*secondary_values*)
Update value.

Parameters **secondary_values** (*str*) – Value to update to.

class kuha_common.document_store.field_types.**Set**(*name*, *value=None*)
Set is a field type with name and list of unique values.

Derived from *Value* Implements methods that differ from parent class.

Parameters

- **name** (*str*) – Name of the field.
- **value** (*list or None*) – Optional value for the field.

set_value(*value*)
Sets value.

Parameters **value** (*list*) – Value for field.

Raises *FieldTypeException* if submitted value is not a list.

add_value(*value*)
Add value to field.

Appends a value to the list of values already set. Makes sure that the list holds no duplicates by silently discarding them.

Parameters **value** (*list or str or None*) – value or values to be appended. If value is None, empties the list.

import_records(*record*)
Import records by adding the submitted records to contained values.

Parameters **record** (*list or str or None*) – Hold the values to be imported.

updates(*secondary_values*)
Updates old values with values contained in this Set.

Looks for combination of secondary_values and values in this set. Discards duplicates and stores the updated values to `value`.

Parameters **secondary_values** (*list*) – list of old values to be updated with new ones.

class kuha_common.document_store.field_types.**Element**(*name*, *attribute_names=None*)
Element is a field type with name, value and attributes.

Derived from *Value*.

Element is used to store fields that contain attributes in addition to a value. Each attribute in itself is an instance of *Value* and is dynamically stored in instance variable `attr_<name>`. When instantiated and populated with values and attributes, the element-instance can be used to get it's value, value's name, but also to get the attributes and their names, even thought the caller does not know the attribute names a priori.

Example of constructing an element (the source):

```
>>> from kuha_common.document_store.field_types import Element
>>> animal = Element('animal', ['color', 'weight', 'height'])
>>> animal.add_value('cat', color='yellow', weight=10, height=5)
```

Example of reading from an unknown element (the receiver):

```
>>> unknown_element.get_name()
'animal'
>>> unknown_element.get_value()
'cat'
>>> for att in unknown_element.iterate_attributes():
...     att.get_name() + ' : ' + str(att.get_value())
...
'height : 5'
'color : yellow'
'weight : 10'
>>> unknown_element.attr_color.get_value()
'yellow'
```

This is especially useful when using as an interchange format. The receiver does not need to know the attribute names beforehand. Instead the receiver can iterate throught every attribute to get their name-value pairs or if the receiver is interested in a single attribute, it may be called by the dynamically constructed instance-variable prefixed with *attr_*.

Parameters

- **name** (*str*) – Name of the field.
- **attribute_names** (*list*) – Optional parameter for attribute names.

Raises *FieldTypeException* if *attribute_names* has duplicates.

is_pending()
Is the element pending for values.

Returns True if pending, False if not.

Return type bool

new()
Create a new element-instance with same name and attributes but without values.

Instantiates a new instance of itself. The new instance is pending for values.

Example:

```
>>> animal = Element('animal', ['color', 'weight', 'height'])
>>> animal.add_value('cat', color='yellow', weight=10, height=5)
>>> another_animal = animal.new()
>>> another_animal.add_value('dog', color='white', weight=30, height=15)
```

Returns new element.

Return type *Element*

add_value(*value=None*, ***attributes*)
Add value with attributes as keyword arguments.

Note This may only be called once for each instance.

Example:

```
>>> from kuha_common.document_store.field_types import Element
>>> animal = Element('animal', ['color', 'weight', 'height'])
>>> animal.add_value('cat', color='yellow', weight=10, height=5)
```

Parameters

- **value** (*str or int or None*) – Value for the element.
- ****attributes** – keyword arguments for attributes of the element.

Raises *FieldTypeException* if the element already has values or if submitted value is None and no attributes are given.

iterate_attributes()
Generator function. Iterates element attributes.

Returns a generator object for iterating attributes.

get_attribute(*name*)
Get attribute by attribute name.

Parameters **name** (*str*) – Name of the attribute to look for.

Returns attribute of the element or None if not found.

Return type *Value* or None

set_attribute(*name*, *value*)
Sets new value for attribute.

Note The element must have an attribute with the *name*.

Parameters

- **name** (*str*) – attribute name.
- **value** (*str or int or None*) – new value.

Raises *FieldTypeException* if element does not have an attribute with submitted *name*.

export_attributes_as_dict()
Export element's attributes as a dictionary.

Returns dictinary representing the attributes

Return type dict

export_dict()
Export the element as a dictionary.

Returns a dictionary with key-value pairs given wrapped inside a another dictionary with the elements key as name.

Example:

```
>>> from kuha_common.document_store.field_types import Element
>>> animal = Element('animal', ['color', 'weight', 'height'])
>>> animal.add_value('cat', color='yellow', weight=10, height=5)
>>> animal.export_dict()
{'animal': {'color': 'yellow', 'weight': 10, 'height': 5, 'animal': 'cat'}}
```

Returns dictinary representing the *Element*

Return type dict

import_records(*record*)
This object does not support importing records.

Raises *FieldTypeException*

updates(*secondary_values*)
Updates attributes not found in this element with the ones found from *secondary_values*.

Parameters **secondary_values** (*dict*) – Attributes from old element.

class kuha_common.document_store.field_types.**LocalizableElement**(*name*, *attribute_names=None*)
LocalizableElement is a field type with name, value, language and attributes.

Derived from *Element*. Has an additional attribute for language. The language is special attribute that is used when updating elements.

Seealso *Element*

Parameters

- **name** (*str*) – Name of the element.
- **attribute_names** (*list*) – Optional list of attribute names.

Raises *FieldTypeException* if *attribute_names* contain a name that is reserved for language.

set_language(*language*)
Set language for element.

Parameters **language** (*str*) – language to set.

Raises *FieldTypeException* if language already set.

get_language()
Get language of element.

Returns language

Return type str or None

add_value(*value=None*, *language=None*, ***attributes*)
Add values for element.

Note This may only be called once for each instance.

Seealso *Element.add_value()*

Parameters

- **value** (*str or int*) – value to set.
- **language** (*str*) – language of the element.
- ****attributes** – keyword arguments for attributes of the element.

Raises TypeError if language is not given or is None.

export_dict()
Export the element as a dictionary.

Seealso *Element.export_dict()*

Returns dictinary representation of the element.

Return type dict

class kuha_common.document_store.field_types.**ElementContainer**(*name*, *sub_element*)
ElementContainer contains a list of single type of Element/LocalizableElement field types.

Receives mandatory parameters for *name* and *sub_element*. The *sub_element* describes the element types that this container can store. Every new element that a container can create will be an instance created from this *sub_element*.

Example:

```
>>> from kuha_common.document_store.field_types import ElementContainer,␣
↪LocalizableElement
>>> animal = LocalizableElement('animal', ['color', 'width', 'height'])
>>> animals = ElementContainer('animals', animal)
>>> animals.add_value('cat', 'en', color='yellow', width=10, height=5)
>>> animals.add_value('kissa', 'fi', color='keltainen', width=10, height=5)
>>> animals.export_dict()  # result formatted for better readability
{'animals': [
    {'width': 10,
     'language': 'en',
     'color': 'yellow',
     'height': 5,
     'animal': 'cat'},
    {'width': 10,
     'language': 'fi',
```

```
     'color': 'keltainen',
     'height': 5,
     'animal': 'kissa'}]
}
```

Elements can be iterated:

```
>>> for animal in animals:
...     animal.attr_color.get_value() + " for language: " + animal.get_language()
...
'yellow for language: en'
'keltainen for language: fi'
```

And updated with containers sharing name and attribute names:

```
>>> another_animal = LocalizableElement('animal', ['color', 'width', 'height'])
>>> more_animals = ElementContainer('animals', another_animal)
>>> more_animals.add_value('dog', 'en', color='white', width=20, height=10)
>>> more_animals.add_value('koira', 'fi', color='valkoinen', width=20, height=10)
>>> animals.updates(more_animals)
>>> animals.export_dict()  # result formatted for better readability
{'animals': [
    {'language': 'en',
     'height': 5,
     'color': 'yellow',
     'animal': 'cat',
     'width': 10},
    {'language': 'fi',
     'height': 5,
     'color': 'keltainen',
     'animal': 'kissa',
     'width': 10},
    {'language': 'en',
     'height': 10,
     'color': 'white',
     'animal': 'dog',
     'width': 20},
    {'language': 'fi',
     'height': 10,
     'color': 'valkoinen',
     'animal': 'koira',
     'width': 20}]
}
```

Parameters

- **name** (*str*) – name of the container.
- **sub_element** (*LocalizableElement* or *Element*) – element to contain.

Raises *FieldTypeException* for invalid sub_element.

import_records(*record*)

Imports records from a list of dictionaries.

Note The dictionaries will lose information.

Parameters **record** (*list*) – list of dictionaries with records to import.

add_value(*value=None*, *language=None*, ***kwargs*)
Add new element to list of elements

Parameters

- **value** (*str or int or None*) – value for the new element.
- **language** (*str or None*) – language for the new element.
- ****kwargs** – key-value pairs for attributes of the new element.

Raises *FieldTypeException* for invalid language parameter depending on whether the `sub_element` is localizable.

export_dict()
Export container as dictionary.

Returns dictionary representing the container.

Return type dict

iterate_values_for_language(*language*)
Generator for iterating contained elements by language.

Parameters **language** (*str*) – language which is used to filter yielded results.

Returns a generator object for iterating elements

get_available_languages()
Get list of languages for this container.

Returns list of distinct languages.

Return type list

updates(*secondary_values*)
Updates contained values with *secondary_values*.

Looks for values that are not currently contained, and appends them as contained values. Also appends different language versions. If a language version has the same value, looks for differences in attributes. If new value has not the same attributes as the old one, adds these attributes to the new value. If old value has same attribute name, it will be discarded.

Note Document Store uses MongoDB as a backend. MongoDB deals with JSON-like objects, which in turn are best represented in Python as dictionaries. The purpose of *kuha_common.document_store.records* is to be used as a global (in Kuha context) interchange format and so it will be best to support both dictionaries and ElementContainers for this operation. Therefore there is some flexibility in the type of parameter that this method accepts.

Note There is a logical difference in which type of parameter is submitted to this method. When using other types than ElementContainers, the parameter's content will be changed.

Parameters **secondary_values** (instance of *ElementContainer* or dict or list) – Old values known to have the same container (must have the same name). If secondary_values is a list, it is assumed that the caller has explicitly checked that the parameter represents old values for this container. Otherwise the name of the container will be checked here and KeyError exceptions will be raised.

class kuha_common.document_store.field_types.**FieldAttribute**(*name*, *parent=None*)
Common attributes for each field type.

Stores fields name, parent fields name and constructs a path for the field. This path can be used when building queries against Document Store. The name can be used to lookup values from objects returned from Document Store.

Used by *FieldTypeFactory* to store information of fields that can be used before the field has been fabricated.

Parameters

- **name** (*str*) – name of the field.
- **parent** (*str*) – optional parameter parent. Used for sub-elements.

value_from_dict(*_dict*)

Get value or values corresponding to `path` from parameter.

Note Returned values cannot be separated by language afterwards.

Parameters **_dict** (*dict*) – dictionary to lookup for `path`.

Returns value or values stored in path of the _dict.

Return type str or list or None

class kuha_common.document_store.field_types.**FieldTypeFactory**(*name*, *sub_name=None*, *attrs=None*, *localizable=True*, *single_value=False*)

Factory for field types.

Stores information for each field, that can be used before the field actually has been initiated. This is useful for building queries against Document Store, because the caller needs to know the names and paths of the fields about to be queried.

The attributes stored here are also used to fabricate each field type. This means that each of the field types supported by *kuha_common.document_store.records* are to be initiated throught this factory.

Seealso *ElementContainer*

Example:

```
>>> from kuha_common.document_store.field_types import FieldTypeFactory
>>> animals_factory = FieldTypeFactory('animals', 'animal', ['color', 'width',
↪'height'])
>>> animals_factory.attr_color.name
'color'
>>> animals_factory.attr_color.path
'animals.color'
>>> animals = animals_factory.fabricate()
>>> animals.add_value('cat', 'en', color='yellow', height=10, width=5)
>>> animals.export_dict()
{'animals': [{'color': 'yellow', 'animal': 'cat', 'height': 10, 'width': 5,
↪'language': 'en'}]}
```

Parameters

- **name** (*str*) – name of the field.
- **sub_name** (*str*) – name of the sub field, if any.
- **attrs** (*list or str*) – field attributes, if any. Multiple attributes in list.

- **localizable** (*bool*) – is the field localizable.
- **single_value** (*bool*) – The fabricated field can contain only a single value.

Raises ValueError if attribute has same name as the element or sub_element.

Raises *FieldTypeException* for parameter combinations that are not supported.

fabricate()
Fabricate field type by factory attributes.

Returns the correct type of field type based on attributes given to the factory at initialization time.

Returns Instance of one of the fields types.

document_store/records.py

Models for records supported by Document Store.

Due to its schemaless design, the document store relies heavily on these models. Use these models when building importers.

kuha_common.document_store.records.**datetime_to_datestamp**(*_datetime*)
Convert datetime object to datestamp string supported by Document Store.

Parameters **datetime** (datetime.datetime) – datetime to convert.

Returns converted datestamp.

Return type str

kuha_common.document_store.records.**datestamp_to_datetime**(*datestamp*)
Convert datestamp string to datetime.datetime object.

Parameters **datestamp** (*str*) – datestamp to convert.

Returns converted datetime.

Return type datetime.datetime

kuha_common.document_store.records.**datetime_now**()
Get current datetime in supported format.

Returns Supported datetime object representing current time.

Return type datetime.datetime

class kuha_common.document_store.records.**RecordBase**(*document_store_dictionary=None*)
Baseclass for each record.

Provides methods used to import, export, create and update records. Dynamically fabricates each class variable of type FieldTypeFactory into an instance variable overriding the class variable.

Note Use this class throught subclasses only.

Parameters **document_store_dictionary** (*dict*) – Optional parameter for creating a record at initialization time. Note that this dictionary will be iterated destructively.

classmethod get_collection()
Get record collection.

Collection is used for queries against the Document Store.

Returns collection of the record.

Return type str

classmethod iterate_record_fields()
Iterate class attributes used as record fields.

Iteration returns tuples: (attribute_name, attribute)

Returns generator for iterating record fields.

export_metadata_dict()
Export record metadata as dictionary.

Returns record's metadata

Return type dict

export_dict(*include_metadata=True*, *include_id=True*)
Return dictionary representation of record.

Parameters

- **include_metadata** (*bool*) – export includes metadata
- **include_id** (*bool*) – export includes id

Returns record

Return type dict

set_updated(*value=None*)
Set updated timestamp.

Sets updated metadata attribute.

Note The timestamp is always stored as `datetime.datetime`, but for convenience it is accepted as a string that is formatted accordingly.

Parameters **value** (`datetime.datetime` or str) – Optional timestamp to set.

set_created(*value=None*)
Set created timestamp.

Sets created metadata attribute.

Note The timestamp is always stored as `datetime.datetime`, but for convenience it is accepted as a string that is formatted accordingly.

Parameters **value** (`datetime.datetime` or str) – Optional timestamp to set.

set_cmm_type(*value=None*)
Set cmm type.

Parameters **value** (*str*) – Optional type to set.

set_id(*value*)
Set ID.

Parameters **value** (*str*) – id to set.

get_updated()
Get updated value.

Note The timestamp is stored as a `datetime.datetime` in `_metadata.attr_updated`, but is returned as a string datestamp when using this method. If there is need to access the `datetime.datetime` object, use `get_value()` of the field.

Returns updated timestamp.

Return type str

get_created()
Get created value.

Note The timestamp is stored as a `datetime.datetime` in `_metadata.attr_created`, but is returned as a string datestamp when using this method. If there is need to access the `datetime.datetime` object, use `get_value()` of the field.

Returns created timestamp.

Return type str

get_id()
Get record ID.

Id comes from the backend storage system.

Returns record ID in storage.

Return type str or None

bypass_update(**fields*)
Add fields to be bypassed on update operation.

Parameters ***fields** (*str*) – fieldnames to bypass.

bypass_create(**fields*)
Add fields to be bypassed on create operation.

Parameters ***fields** (*str*) – fieldnames to bypass.

updates_record(*old_record_dict*)
Update record by appending old values that are not present in current record. Use old record's _id and _metadata.created if present.

Note parameter is a dictionary since MongoDB returns records as JSON-like objects, which in turn are best represented as dictionaries in python.

Parameters **old_record_dict** (*dict*) – Old record as a dictionary.

updates(*secondary_record*)
Update record by appending values from secondary which are not present in this record.

Parameters **secondary_record** (Record instance subclassed from *RecordBase*) – lookup values from this record.

class kuha_common.document_store.records.**Study**(*study_dict=None*)
Study record.

Derived from *RecordBase*. Used to store and manipulate Study records. Study number is a special attribute and it cannot be updated.

All attributes of the record are declared as class variables initiated from *kuha_common.document_store.field_types.FieldTypeFactory*. Instance methods defined in this class are used to add/set values to record attributes. The signatures of the methods are dynamically constructed by the definition of the FieldTypeFactory instances. If, for example, there is a class variable definition:

```
animals = FieldTypeFactory('animals', 'animal', ['color', 'weight', 'height'])
```

The correct method signature should be:

```
def add_animals(self, value, language, color=None, weight=None, height=None):
```

For the dynamic nature of the record-model these signatures are left open, and python's *args and **kwargs are used instead. Note that the field type used will raise exceptions if keyword argument key is not found in the initial definition of the field type.

Create a new study record:

```
>>> study = Study()
>>> study.add_study_number(1234)
>>> study.add_study_titles('Study about animals', 'en')
>>> study.add_principal_investigators('investigator', 'en', organization='Big␣
↪organization ltd.')
```

Import existing study record from dictionary:

```
>>> study_dict = {'study_number': 1234,
... 'study_titles': [{'study_title': 'Study about animals', 'language': 'en'}],
... 'principal_investigators': [{'principal_investigator': 'investigator',
... 'language': 'en', 'organization': 'Big organization ltd.'}]}
>>> study = Study(study_dict)
```

Iterate attributes:

```
>>> for pi in study.principal_investigators:
...     pi.attr_organization.get_value()
...
'Big organization ltd.'
```

Seealso *RecordBase* and *kuha_common.document_store.field_types*

Parameters **study_dict** (*dict*) – Optional study record as dictionary used for constructing a record instance.

study_number = <kuha_common.document_store.field_types.FieldTypeFactory object>
Study number is used to identify a study. It must be unique within records, not localizable and contain only a single value. It cannot be updated.

persistent_identifiers = <kuha_common.document_store.field_types.FieldTypeFactory obje
Persistent identifiers. Multivalue-field with unique values.

identifiers = <kuha_common.document_store.field_types.FieldTypeFactory object>
Identifiers. Localizable field with agency-attribute. This needs to be localizable for the sake of agency-attribute. Note that two identical identifiers with same locale cannot exists at same time. The latter agency will overwrite the former on update.

study_titles = <kuha_common.document_store.field_types.FieldTypeFactory object>
Study titles. Localizable, multivalue-field without attributoes.

document_titles = <kuha_common.document_store.field_types.FieldTypeFactory object>
Document titles. Localizable, multivalue-field without attributoes.

parallel_titles = <kuha_common.document_store.field_types.FieldTypeFactory object>
Parallele study titles. Localizable, multivalue-field without attributes.

principal_investigators = <kuha_common.document_store.field_types.FieldTypeFactory obj
Pricipal investigators. Localizable, multivalue-field with organization-attribute.

publishers = <kuha_common.document_store.field_types.FieldTypeFactory object>
Publishers. Localizable, multivalue-field with abbreviation-attribute.

distributors = <kuha_common.document_store.field_types.FieldTypeFactory object>
Distributors. Localizable, multivalue-field with abbreviation and uri attributes.

document_uris = <kuha_common.document_store.field_types.FieldTypeFactory object>
Document URIs. Localizable, multivalue-field with location and description attributes.

study_uris = <kuha_common.document_store.field_types.FieldTypeFactory object>
Study URIs. Localizable, multivalue-field with location and description attributes.

publication_dates = <kuha_common.document_store.field_types.FieldTypeFactory object>
Publication dates. Localizable, multivalue-field without attributes. Note that these are treated as strings, not datetime-objects.

publication_years = <kuha_common.document_store.field_types.FieldTypeFactory object>
Publication years. Localizable, multivalue-field with distribution date attribute.

abstract = <kuha_common.document_store.field_types.FieldTypeFactory object>
Abstract. Localizable, multivalue-field.

classifications = <kuha_common.document_store.field_types.FieldTypeFactory object>
Classifications. Localizable, multivalue-field with system name, uri and description attributes.

keywords = <kuha_common.document_store.field_types.FieldTypeFactory object>
Keywords. Localizable, multivalue-field with system name, uri and description attributes.

time_methods = <kuha_common.document_store.field_types.FieldTypeFactory object>
Time methods. Localizable, multivalue-field with system name, uri and description attribute.

sampling_procedures = <kuha_common.document_store.field_types.FieldTypeFactory object>
Sampling procedures. Localizable, multivalue-field with description, system name and uri attibutes.

collection_modes = <kuha_common.document_store.field_types.FieldTypeFactory object>
Collection modes. Localizable, multivalue-field with system name and uri attritubes.

analysis_units = <kuha_common.document_store.field_types.FieldTypeFactory object>
Analysis units. Localizable, multivalue-field with system name, uri and description attributes.

collection_periods = <kuha_common.document_store.field_types.FieldTypeFactory object>
Collection periods. Localizable, multivalue-field with event-attribute.

data_kinds = <kuha_common.document_store.field_types.FieldTypeFactory object>
Data kinds. Localizable, multivalue-field.

study_area_countries = <kuha_common.document_store.field_types.FieldTypeFactory object
Study area countries. Localizable, multivalue-field with abbreviation attribute.

geographic_coverages = <kuha_common.document_store.field_types.FieldTypeFactory object
Geographic coverages. Localizable, multivalue-field.

universes = <kuha_common.document_store.field_types.FieldTypeFactory object>
Universes. Localizable, multivalue-field with included attribute.

data_access = <kuha_common.document_store.field_types.FieldTypeFactory object>
Data access. Localizable, multivalue-field.

data_access_descriptions = <kuha_common.document_store.field_types.FieldTypeFactory ob
Data access descriptions. Localizable, multivalue-field.

citation_requirements = <kuha_common.document_store.field_types.FieldTypeFactory objec
Citation requirements. Localizable, multivalue-field.

deposit_requirements = <kuha_common.document_store.field_types.FieldTypeFactory object
Deposit requirements. Localizable, multivalue-field.

file_names = <kuha_common.document_store.field_types.FieldTypeFactory object>
File names. Localizable, multivalue-field.

instruments = <kuha_common.document_store.field_types.FieldTypeFactory object>
Instruments. Localizable, multivalue-field with instrument name attribute.

related_publications = <kuha_common.document_store.field_types.FieldTypeFactory object
Related publications. Localizable multivalue-field

study_groups = <kuha_common.document_store.field_types.FieldTypeFactory object>
Study groups. Localizable, multivalue-field with name and description attributes.

copyrights = <kuha_common.document_store.field_types.FieldTypeFactory object>
Copyrights. Localizable, multivalue-field.

data_collection_copyrights = <kuha_common.document_store.field_types.FieldTypeFactory
Copyrights. Localizable, multivalue-field.

collection = 'studies'
Database collection (table) for persistent storage.

cmm_type = 'study'
CMM type for Study.

add_study_number(*value*)
Add study number.

Note despite the name, the value does not need to be a number.

Parameters **value** (*str or int*) – study number.

add_persistent_identifiers(*value*)
Add persistent identifiers

Parameters **value** (*str or int*) – persistent identifier

add_identifiers(*value*, **args*, ***kwargs*)
Add identifiers.

Parameters

- **value** (*str or int*) – identifier
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_study_titles(*value*, **args*, ***kwargs*)
Add study titles.

Parameters

- **value** (*str*) – study title.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_document_titles(*value*, **args*, ***kwargs*)
Add document titles.

Parameters

- **value** (*str*) – document title.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_parallel_titles(*value*, **args*, ***kwargs*)
Add parallel titles.

Parameters

- **value** (*str*) – title.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_principal_investigators(*value*, **args*, ***kwargs*)
Add principal investigators.

Parameters

- **value** (*str*) – investigators.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_publishers(*value*, **args*, ***kwargs*)
Add publishers.

Parameters

- **value** (*str*) – publishers.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_distributors(*value*, **args*, ***kwargs*)
Add distributors.

Parameters

- **value** (*str*) – distributor.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_document_uris(*value*, **args*, ***kwargs*)
Add document URIs.

Parameters

- **value** (*str*) – URI.

- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_study_uris(*value*, **args*, ***kwargs*)
Add study URIs.

Parameters

- **value** (*str*) – URI.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_publication_dates(*value*, **args*, ***kwargs*)
Add publication dates.

Parameters

- **value** (*str*) – date.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_publication_years(*value*, **args*, ***kwargs*)
Add publication dates.

Parameters

- **value** (*str*) – date.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_abstract(*value*, **args*, ***kwargs*)
Add abstract.

Parameters

- **value** (*str*) – abstract.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_classifications(*value*, **args*, ***kwargs*)
Add classifications.

Parameters

- **value** (*str*) – classifications.

- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_keywords(*value*, **args*, ***kwargs*)
Add keywords.

Parameters

- **value** (*str*) – keyword.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_time_methods(*value*, **args*, ***kwargs*)
Add time methods.

Parameters

- **value** (*str*) – time method
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_sampling_procedures(*value*, **args*, ***kwargs*)
Add sampling procedures

Parameters

- **value** (*str*) – sampling procedure
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_collection_modes(*value*, **args*, ***kwargs*)
Add collection modes

Parameters

- **value** (*str*) – collection mode
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_analysis_units(*value*, **args*, ***kwargs*)
Add analysis units.

Parameters

- **value** (*str*) – analysis unit.

- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_collection_periods(*value*, **args*, ***kwargs*)
Add collection periods.

Parameters

- **value** (*str*) – collection period.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_data_kinds(*value*, **args*)
Add data kinds.

Parameters

- **value** (*str*) – data kind.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_study_area_countries(*value*, **args*, ***kwargs*)
Add study area countries.

Parameters

- **value** (*str*) – country.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_geographic_coverages(*value*, **args*)
Add geographic coverages

Parameters

- **value** (*str*) – geographic coverage
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_universes(*value*, **args*, ***kwargs*)
Add universes.

Parameters

- **value** (*str*) – universe.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_data_access(*value*, **args*, ***kwargs*)
Add data access.

Parameters

- **value** (*str*) – data access.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_data_access_descriptions(*value*, **args*, ***kwargs*)
Add data access descriptions.

Parameters

- **value** (*str*) – access description.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_citation_requirements(*value*, **args*)
Add citation requirements.

Parameters

- **value** (*str*) – citation requirement.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_deposit_requirements(*value*, **args*)
Add deposit requirements.

Parameters

- **value** (*str*) – deposit requirement.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_file_names(*value*, **args*, ***kwargs*)
Add file name.

Parameters

- **value** (*str*) – file name.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_instruments(*value*, **args*, ***kwargs*)
Add instrument.

Parameters

- **value** (*str*) – instrument.

- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_related_publications(*value*, **args*, ***kwargs*)
Add related publications.

Parameters

- **value** (*str*) – publication.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_study_groups(*value*, **args*, ***kwargs*)
Add study group.

Parameters

- **value** (*str*) – study group.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_copyrights(*value*, **args*, ***kwargs*)
Add copyright.

Parameters

- **value** (*str*) – copyright.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_data_collection_copyrights(*value*, **args*)
Add data collection copyrights.

Parameters

- **value** (*str*) – data collection copyright.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

updates(*secondary*)
Check that records have common unique keys. Update record by appending values from secondary which are not present in this record.

Parameters **secondary** (*Study*) – Lookup values to update from secondary record.

Returns True if record updated, False if not.

Return type bool

class kuha_common.document_store.records.**Variable**(*variable_dict=None*)
Variable record.

Derived from *RecordBase*. Used to store and manipulate variable records. Study number and variable name are special attributes and cannot be updated.

Seealso *Study* documentation for more information.

Parameters **variable_dict** (*dict*) – Optional variable record as dictionary used for constructing a record instance.

study_number = <kuha_common.document_store.field_types.FieldTypeFactory object>
Study number and variable name are used to identify a variable within variable records. Their combination must be unique withing variable records, they cannot be localizable, and they can only contain a single value. They also cannot be updated.

variable_name = <kuha_common.document_store.field_types.FieldTypeFactory object>
Variable name within a study. See also *study_number*

question_identifiers = <kuha_common.document_store.field_types.FieldTypeFactory object
Question identifiers, if variable refers to a question. Not localizable, multiple unique values.

variable_labels = <kuha_common.document_store.field_types.FieldTypeFactory object>
Variable labels. Localizable, multivalue-field.

codelist_codes = <kuha_common.document_store.field_types.FieldTypeFactory object>
Codelist codes. Localizable, multivalue-field with label and missing attributes.

collection = 'variables'
Database collection for persistent storage.

cmm_type = 'variable'
CMM type for variable.

add_study_number(*value*)
Add study number.

Parameters **value** (*str or int.*) – study number.

add_variable_name(*value*)
Add variable name.

Parameters **value** (*str*) – variable name.

add_question_identifiers(*value*)
Add question identifier

Parameters **value** (*str or int.*) – question identifier.

add_variable_labels(*value*, **args*, ***kwargs*)
Add variable label

Parameters

- **value** (*str*) – variable label.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_codelist_codes(*value*, **args*, ***kwargs*)
Add codelist code

Parameters

- **value** (*str*) – codelist code.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

updates(*secondary*)
Check that records have common unique keys. Update record by appending values from secondary which are not present in this record.

Parameters **secondary** (*Variable*) – Lookup values to update from secondary record.

Returns True if record updated, False if not.

Return type bool

class kuha_common.document_store.records.**Question**(*question_dict=None*)
Question record.

Derived from *RecordBase*. Used to store and manipulate question records. *study_number* and question_idenntifier are special attributes and cannot be updated.

Seealso *Study* documentation for more information.

Parameters **question_dict** (*dict*) – Optional question record as dictionary used for constructing a record instance.

study_number = <kuha_common.document_store.field_types.FieldTypeFactory object>
Study number and question identifier are used to identify a question. Their combination must be unique withing records, they must not be localizable and they can only contain a single value. They also cannot be updated.

question_identifier = <kuha_common.document_store.field_types.FieldTypeFactory object>
Question identifier within a study. See also *study_number*

variable_name = <kuha_common.document_store.field_types.FieldTypeFactory object>
Variable name that specifies the variable for the question. Not localizable, single value.

question_texts = <kuha_common.document_store.field_types.FieldTypeFactory object>
Question texts. Localizable, multivalue-field.

research_instruments = <kuha_common.document_store.field_types.FieldTypeFactory object
Research instruments. Localizable, multivalue-field.

codelist_references = <kuha_common.document_store.field_types.FieldTypeFactory object>
Codelist references. Localizable, multivalue-field.

collection = 'questions'
Database collection for persistent storage.

cmm_type = 'question'
CMM type for question

add_study_number(*value*)
Add study number.

Parameters **value** (*str or int.*) – study number.

add_question_identifier(*value*)
Add question identifier

Parameters **value** (*str or int.*) – question identifier.

add_variable_name(*value*)
Add variable name.

Parameters **value** (*str*) – variable name.

add_question_texts(*value*, **args*, ***kwargs*)
Add question text

Parameters

- **value** (*str*) – question text.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_research_instruments(*value*, **args*, ***kwargs*)
Add research instrument

Parameters

- **value** (*str*) – research instrument.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_codelist_references(*value*, **args*, ***kwargs*)
Add codelist reference

Parameters

- **value** (*str*) – codelist reference
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

updates(*secondary*)
Check that records have common unique keys. Update record by appending values from secondary which are not present in this record.

Parameters **secondary** (*Question*) – Lookup values to update from secondary record.

Returns True if record updated, False if not.

Return type bool

class kuha_common.document_store.records.**StudyGroup**(*study_group_dict=None*)
Study group record.

Derived from *RecordBase*. Used to store and manipulate study group records. *study_group_identifier* is a arpecial attribute and cannot be updated.

Seealso *Study* documentation for more information.

Parameters **study_group_dict** (*dict*) – Optional study group record as dictionary used for constructing a record instance.

study_group_identifier = <kuha_common.document_store.field_types.FieldTypeFactory obje
Study group identifier. Used to identify study group. Must be unique within study groups, cannot be localizable and can contain only a single value. This value cannot be updated.

study_group_names = <kuha_common.document_store.field_types.FieldTypeFactory object>
Study group names. Localizable, multivalue-field.

descriptions = <kuha_common.document_store.field_types.FieldTypeFactory object>
Study group descriptions. Localizable, multivalue-field.

uris = <kuha_common.document_store.field_types.FieldTypeFactory object>
Study group URIs. Localizable, multivalue-field.

study_numbers = <kuha_common.document_store.field_types.FieldTypeFactory object>
Study numbers. Multivalue-field with unique values.

collection = 'study_groups'
Database collection for persistent storage.

cmm_type = 'study_group'
CMM type for study groups.

add_study_group_identifier(*value*)
Add study group identifier.

Parameters **value** (*str or int*) – Study group identifier.

add_study_group_names(*value*, **args*, ***kwargs*)
Add study group names

Parameters

- **value** (*str*) – study group name.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*
- ****kwargs** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_descriptions(*value*, **args*)
Add study group descriptions

Parameters

- **value** (*str*) – study group description.
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_uris(*value*, **args*)
Add study group URIs

Parameters

- **value** (*str*) – study group URI
- ***args** – defined by the parameters given to *kuha_common.document_store.field_types.FieldTypeFactory*

add_study_numbers(*value*)
Add study number.

Parameters **value** (*str or int*) – study number.

updates (*secondary*)
Check that records have common unique keys. Update record by appending values from secondary which are not present in this record.

Parameters **secondary** (*StudyGroup*) – Lookup values to update from secondary record.

Returns True if record updated, False if not.

Return type bool

kuha_common.document_store.records.**record_factory**(*ds_record_dict*)
Dynamically construct record instance based on given document store dictionary.

Looks up the correct record by the cmm type found from *ds_record_dict* metadata.

Parameters **ds_record_dict** (*dict*) – record received from Document Store.

Returns Record instance.

Return type *Study* or *Variable* or *Question* or *StudyGroup*

kuha_common.document_store.records.**record_by_collection**(*collection*)
Finds a record class by the given collection.

Parameters **collection** (*str*) – collection of the record.

Returns record class

Return type *Study* or *Variable* or *Question* or *StudyGroup*

Raises KeyError if collection is not found in any record.

document_store/mappings

document_store/mappings/exceptions.py

Exceptions for mapping package.

exception kuha_common.document_store.mappings.exceptions.**InvalidMapperParameters**
Raise for invalid configuration of a mapper - Coding error.

For example trying to add attributes to a mapper which does not support attributes. These errors are coding errors and should be treated as such.

exception kuha_common.document_store.mappings.exceptions.**MappingError**
Raise for errors while mapping input - User error.

Subclass to create more precise error classes.

exception kuha_common.document_store.mappings.exceptions.**ParseError**
Unable to parse source XML.

Note Mask ElementTree ParseError so caller may use MappingError to catch all user-errors when mapping.

exception kuha_common.document_store.mappings.exceptions.**UnknownXMLRoot**(*expected=None*, *unknown=None*)
Unknown root element.

exception kuha_common.document_store.mappings.exceptions.**MissingRequiredAttribute**(**xpaths*, *msg=None*)
Source does not contain a required attribute.

exception kuha_common.document_store.mappings.exceptions.**InvalidContent**
Attribute found but contains invalid data.

document_store/mappings/xmlbase.py

Components to use for XML parsing & mapping to Document Store records.

Contains a base class to use for parsing XML to Document Store records. Provides common functions useful in parsing XML data.

class kuha_common.document_store.mappings.xmlbase.**MappedParams**(*value*)
Contains parameters ready to pass to record's add-methods.

XMLMapper retrieves parameters from XML record and stores them in an instance of this class. The record instances add-methods get called with stored parameters by using tuple and dict unpacking.

Example:

```
mapped_params = MappedParams('study_identifier')
mapped_params.set_language('en')
mapped_params.keyword_arguments.update({'agency': 'archive'})
study = Study()
study.add_identifiers(*mapped_params.arguments, **mapped_params.keyword_arguments)
```

Parameters **value** (*str or None*) – value used as the first argument

has_language()
True if MappedParams has language argument

Returns True if has language, False if not.

Return type bool

set_language(*language*)
Set language argument. Will overwrite if previously set.

Parameters **language** (*str*) – Language to set.

get_language()
Get language argument.

Returns Language

Return type str

get_value()
Get value argument.

Returns value.

Return type str or None

copy()
Make a copy of the object with contents and return the copy.

Returns copy of this *MappedParams*

Return type *MappedParams*

has_arguments()
Return True if *MappedParams* has arguments or keyword_arguments.

Returns True if object has arguments or keyword_arguments.

Return type bool

class kuha_common.document_store.mappings.xmlbase.**XMLMapper**(*xpath*, *from_attribute=None*, *required=False*, *localizable=True*)

XMLMapper populates *MappedParams* instances from XML.

Parameters

- **xpath** (*str*) – XPath where to look for element containing value.
- **from_attribute** (*str or None*) – Look value from attribute of the element.
- **required** (*bool*) – raises MissingRequiredAttribute if value is not found.
- **localizable** (*bool*) – True if the value is localizable, False if not.

set_value_conversion(*conv_func*)

Set conversion callable.

Note *conv_func* must accept a string or None as a parameter and return the converted value.

Parameters **conv_func** (*callable.*) – Callable used for conversion.

Returns self

set_value_getter(*getter_func*)

Set value getter callable.

Note *getter_func* must accept an XML element xml.etree.ElementTree.Element as a parameter and return the value.

Parameters **getter_func** (*callable.*) – Callable used for getting a value from XML element.

Returns self

expect_single_value()

This mapper will be expected to return a single value.

Returns self

expect_multiple_values()

This mapper will be expected to return multiple values.

Returns self

disable_attributes()

This mapper will not contain any attributes.

Returns self

iterate_attributes(**relations*)

Iterate attributes to map.

Parameters ***relations** (*str*) – optional parameters to iterate only attributes with a certain relation.

Returns A generator yielding tuples of each attribute in the format: (attribute_name, attribute_mapper, attribute_provides_main_lang)

Return type generator

as_params(*element*, *default_language*, *xml_namespaces*)

Use mapping to construct a *MappedParams* from XML element.

Use mapper's `_value_getter` and `_value_conversion` to get value from XML element. Construct a *MappedParams* from the value. If mapping `localizable` is True add language from XML elements xml:lang attribute.

Parameters

- **element** (`xml.etree.ElementTree.Element`) – XML element.
- **default_language** (*str*) – default language if element has none.
- **xml_namespaces** (*dict*) – XML namespaces for the element.

Returns mapped parameters ready to pass to records add-method.

Return type *MappedParams*

add_attribute(*att_name*, *mapper*, *relative=True*, *provides_main_lang=False*)
Add attribute to mapper.

Counts the correct xpath if attribute's mapper's xpath is a parent element (starting with '/..'). Includes all needed information of the attribute to a list of tuples contained in `attributes`.

Parameters

- **att_name** (*str*) – attribute name
- **mapper** (*XMLMapper*) – mapper instance for mapping value for the attribute.
- **relative** (*bool*) – Is the attribute map's xpath relative to this element. Defaults to True.
- **provides_main_lang** (*bool*) – Should the language of the attribute be used as a language when mapping this value. Defaults to False.

Raises `InvalidMapperParameters` for conflicting parameters such as: 1. Calling this method on a mapper which has disabled use of attributes. 2. Using `provides_main_lang` for a non-localizable mapper. 3. Setting `relative` to False on a mapper whose `xpath` refers to parent element.

Returns self

value_params(*source_xml_element*, *default_language*, *xml_namespaces*, *position=None*)
Generate sinle *MappedParams* object from source XML.

Parameters

- **source_xml_element** (`xml.etree.ElementTree.Element`) – XML element.
- **default_language** (*str*) – Default language.
- **xml_namespaces** (*dict*) – XML namespaces.
- **position** (*int or None*) – Optional position for parent xpaths.

Returns generator yielding *MappedParams*.

Return type generator

Raises `MissingRequiredAttribute` if mapper's `required` is True, but xpath provides no element or the element provides no value.

values_params(*source_xml_element*, *default_language*, *xml_namespaces*)
Generate multiple *MappedParams* objects from source XML.

The generated MappedParams will contain attributes as keyword_arguments.

Parameters

- **source_xml_element** (xml.etree.ElementTree.Element) – XML element.
- **default_language** (*str*) – Default language.
- **xml_namespaces** (*dict*) – XML namespaces.

Returns generator yielding *MappedParams*.

Return type generator

Raises MissingRequiredAttribute if mapper's required is True, but xpath provides no element or the element provides no value.

class kuha_common.document_store.mappings.xmlbase.**XMLParserBase**(*root_element*)

Base class where parsers get derived from.

Declares the public API to be used in callers.

Input:

- from_file()
- from_string()

Output:

- studies
- variables
- questions
- study_groups
- all
- select(collection=None)

Provides common functionality to be used within subclasses which map XML-data to Document Store records. Subclasses must implement necessary generators that generate document store records.

Use in subclass:

```
class XMLRecordParser(XMLParserBase):
    @property
    def studies(self):
        maps = [(Study.add_study_number, self._map_single(xpath_to_study_number,␣
↪required=True)),
                (Study.add_study_titles, self._map_multi(xpath_to_study_title))]
        for study_element in self.root_element.findall(xpath_to_study_element,␣
↪self.NS):
            study = Study()
            self._map_to_record(study, study_element, maps)
            yield study
```

Parameters **root_element** (xml.etree.ElementTree.Element) – XML root.

NS = {'xsi': 'http://www.w3.org/2001/XMLSchema-instance', 'xml': 'http://www.w3.org/

XML namespaces

default_language = 'en'

Default language.

classmethod from_string(*xml_body*)

Get parser that iteratively parses XML and generates populated Document Store record instances.

Parameters **xml_body** (*str*) – XML Document as a string. This may come directly from HTTP request body.

Returns parser for iteratively parsing XML and generating Document Store records.

Return type *XMLParserBase*

classmethod from_file(*filepath*)
Get parser that iteratively parses XML and generates populated Document Store record instances.

Parameters **filepath** (*str*) – Path for the XML file.

Returns parser for iteratively parsing XML and generating Document Store records.

Return type *XMLParserBase*

classmethod child_text(*xpath*)
Returns a function which will lookup a child element from given xpath. The returned function takes a single element as a parameter which should be an `xml.etree.ElementTree.Element` or similar. When executed the function returns the child element's text contents or None if child element cannot be found.

Parameters **xpath** – xpath to child. relative to parent.

Returns function which accepts the parent element as a parameter.

Return type function

root_element
Get root element.

Returns Root element

Return type `xml.etree.ElementTree.Element`

root_language
Get language of the root element. If root does not have a language, returns `self.default_language`.

Returns root element language.

Return type str

study_number
Get study number as formatted in source XML.

Seealso `self.study_number_identifier`

Returns Study number from source XML.

Return type str

study_number_identifier
Get study number converted as a valid Document Store identifier.

Returns Study number as valid Document Store identifier.

Return type str

studies
Studies generator. Must be implemented in subclass.

Returns Generator which yields Document Store studies.

variables
Variables generator. Must be implemented in subclass.

Returns Generator which yields Document Store variables.

questions
Questions generator. Must be implemented in subclass.

Returns Generator which yields Document Store questions.

study_groups
Study groups generator. Must be implemented in subclass.

Returns Generator which yields Document Store study groups.

all
Iterate all records found from source XML.

Returns Generator which yields Document Store records.

Return type Generator

select(*collection=None*)
Returns a selective parser. Call with a Document Store collection as parameter to select records only for certain collection.

Note: The returned attributes are defined in subclasses, so they may or may not be generators.

Parameters **collection** (*str or None*) – Document Store collection to select only records belonging to this collection.

Returns Generator which yields Document Store records.

Return type Generator

kuha_common.document_store.mappings.xmlbase.**as_valid_identifier**(*candidate*)
Convert candidate to a string that conforms the rules of validation.

Indentifier must match regex: [a-zA-Z0-9]+[a-zA-Z0-9?_()-.]*’”]

Note: Regex is defined in Document Store. Should it be moved to kuha_common?

Returns identifier which conforms the rules of validation.

Return type str

kuha_common.document_store.mappings.xmlbase.**str_equals**(*correct*, *default=None*)
Conversion function wrapper to compare strings for equality.

Wrapper function that formats comparison value and default value for returned comparison function.

Check if string found from element value or element attribute equals to *correct*.

Parameters

- **correct** (*str*) – comparison string.
- **default** (*str*) – If the value parameter of the comparison function is None, return this value.

Returns function which accepts a single parameter for comparison. Returns True or False, or *default* if the parameter is None.

Return type function

kuha_common.document_store.mappings.xmlbase.**fixed_value**(*fixed*)
Fixed value.

Parameters **fixed** – Use this value

Returns function which accepts a single argument value. The function always returns fixed.

Return type function

kuha_common.document_store.mappings.xmlbase.**element_remove_whitespaces**(*element*)
Conversion function to remove extra whitespace from end of element text.

Iterates element's inner text using `xml.etree.ElementTree.Element.itertext()` which iterates over this element and all subelements. Removes extra whitespaces so paragraphs of text will only have one separating whitespace character.

Parameters **element** (`xml.etree.ElementTree.Element`) – Element from which to get text.

Returns Element's inner text without extra whitespace.

Return type str

kuha_common.document_store.mappings.xmlbase.**element_strip_descendant_text**(*element*)
Conversion function to remove inner elements and their contents.

Parameters **element** (`xml.etree.ElementTree.Element`) – Element for lookup.

Returns Element's inner text without text from descendants and without extra whitespace.

Return type str

document_store/mappings/ddi.py

Mapping profiles for DDI.

Note: has strict dependency to *kuha_common.document_store.records*

---

class kuha_common.document_store.mappings.ddi.**DDI122RecordParser**(*root_element*)
Parse Document Store records from DDI 1.2.2. XML.

studies
Parse XML to create and populate *kuha_common.document_store.records.Study*.

Returns Generator to Populate Document Store Study record.

variables
Parse XML to create and populate multiple *kuha_common.document_store.records.Variable* instances.

Returns Generator to populate multiple Document Store Variable records.

questions
Parse XML to create and populate multiple *kuha_common.document_store.records.Question* instances.

Returns Generator to populate multiple Document Store Question records.

study_groups
Parse XML to create and populate multiple *kuha_common.document_store.records.StudyGroup* instances.

Returns Generator to populate multiple Document Store StudyGroup records.

class kuha_common.document_store.mappings.ddi.**DDI25RecordParser**(*root_element*)
Parse Document Store records from DDI 2.5 XML.

NS = {'xsi': 'http://www.w3.org/2001/XMLSchema-instance', 'xml': 'http://www.w3.org/
XML namespaces

class kuha_common.document_store.mappings.ddi.**DDI31RecordParser**(*root_element*)
Parse Document Store records from DDI 3.1. XML

Check the root element. Expects either ddi:DDIInstance or s:StudyUnit. Currently supports only single s:StudyUnit element within the root.

Parameters **root_element** (xml.etree.ElementTree.Element) – XML root element.

Raises UnknownXMLRoot for unexpected root element.

Raises MappingError if root contains more or less that exactly one s:StudyUnit child.

NS = {'dc': 'ddi:datacollection:3_1', 'l': 'ddi:logicalproduct:3_1', 'ddi': 'ddi:in
XML namespaces

studies
Parse XML to create and populate *kuha_common.document_store.records.Study*.

Returns Generator to Populate Document Store Study record.

variables
Parse XML to create and populate *kuha_common.document_store.records.Variable*.

Returns Generator to Populate Document Store Variable records.

questions
Parse XML to create and populate *kuha_common.document_store.records.Question*.

Returns Generator to Populate Document Store Question records.

study_groups
Parse XML to create and populate *kuha_common.document_store.records.StudyGroup*.

Returns Generator to Populate Document Store StudyGroup records.

testing

Package for common testing functions and classes.

kuha_common.testing.**time_me**(*func*)
Decorate function to print its execution time to stdout.

Note test runner may capture the output.

Parameters **func** – Function to decorate. Count execution time of function.

kuha_common.testing.**mock_coro**(*dummy_rval=None*, *func=None*)
Mock out a coroutine function.

Accepts either keyword argument but not both. Submitting both will raise TypeError.

Mock out coroutine and set return value:

```
>>> coro = mock_coro(dummy_rval='return_value')
>>> rval = await coro()
>>> assert rval == 'return_value'
```

Mock out coroutine with custom function:

```
>>> call_args = []
>>> async def custom_func(*args):
>>>     call_args.append(args)
>>> coro = mock_coro(func=custom_func)
>>> await coro()
>>> assert call_args == [('expected', 'args')]
```

Use as a side_effect when patching:

```
>>> @mock.patch.object(pkg.Class, 'async_method', side_effect=mock_coro())
>>> def test_something(mock_method):
>>>     inst = pkg.Class()
>>>     eventloop.run_until_complete(inst.async_method())
>>>     mock_method.assert_called_once_with()
```

Parameters

- **dummy_rval** – return value of dummy function.
- **func** – function to call instead of original mocked out function.

Returns coroutine function.

kuha_common.testing.**MockCoro**(*dummy_rval=None*, *func=None*)

Mock out a coroutine function.

Accepts either keyword argument but not both. Submitting both will raise TypeError.

Mock out coroutine and set return value:

```
>>> coro = mock_coro(dummy_rval='return_value')
>>> rval = await coro()
>>> assert rval == 'return_value'
```

Mock out coroutine with custom function:

```
>>> call_args = []
>>> async def custom_func(*args):
>>>     call_args.append(args)
>>> coro = mock_coro(func=custom_func)
>>> await coro()
>>> assert call_args == [('expected', 'args')]
```

Use as a side_effect when patching:

```
>>> @mock.patch.object(pkg.Class, 'async_method', side_effect=mock_coro())
>>> def test_something(mock_method):
>>>     inst = pkg.Class()
>>>     eventloop.run_until_complete(inst.async_method())
>>>     mock_method.assert_called_once_with()
```

Parameters

- **dummy_rval** – return value of dummy function.
- **func** – function to call instead of original mocked out function.

Returns coroutine function.

testing/testcases.py

Test cases for Kuha

class kuha_common.testing.testcases.**KuhaUnitTestCase**(*methodName='runTest'*)
Base class for unittests.

- Assertion methods to check record equality.
- Helper methods to provide access to dummydata.

dummydata_dir = '/home/docs/checkouts/readthedocs.org/user_builds/kuha2/envs/0.x.x/src
Override in sublass to lookup dummydata from different directory.

classmethod get_dummydata_path(*path*)
Get absolute path to dummydatafile

Parameters **path** – Path. Gets turned into an absolute if it isn't

Returns Absolute path.

Return type str

classmethod get_dummydata(*path*)
Get dummydata by reading file from `path`

Parameters **path** – path to file.

Returns Contents of the file.

classmethod remove_dummyfile_if_exists(*path*)
Remove dummyfile from `path` if it exists.

Parameters **path** – path to dummyfile.

Returns None

classmethod set_val(*value*)
Assign value as dummyvalue.

Parameters **value** – Value to assign

Returns value

classmethod gen_val(*lenght=None*, *unique=False*, *chars=None*)
Generate & assign dummyvalue.

Parameters

- **lenght** (*int or None*) – lenght of the value
- **unique** (*bool*) – should the value be unique
- **chars** (*str or None.*) – use specific characters.

Returns generated value

Return type str

classmethod gen_id()
Generate Id.

Returns Generated id.

Return type str

classmethod generate_dummy_study()
Generate and return a Study with dummydata.

Returns study with dummydata

Return type *kuha_common.document_store.records.Study*

classmethod generate_dummy_variable()
Generate and return a Variable with dummydata.

Returns variable with dummydata

Return type *kuha_common.document_store.records.Variable*

classmethod generate_dummy_question()
Generate and return a Question with dummydata.

Returns question with dummydata

Return type *kuha_common.document_store.records.Question*

classmethod generate_dummy_studygroup()
Generate and return a StudyGroup with dummydata.

Returns studygroup with dummydata.

Return type *kuha_common.document_store.records.StudyGroup*

setUp()
Format testcase values and initialize event loop.

Call asynchronous code synchronously:

```
self._loop.run_until_complete(coro())
```

tearDown()
Stop patchers.

await_and_store_result(*coro*)
Await coroutine and store returning result.

Example:

```
self._loop.run_until_complete(self.await_future_and_store_result(coro()))
```

Parameters **coro** – Coroutine or Future to await

init_patcher(*patcher*)
Initialize patcher, store for later use, return it.

Parameters **patcher** (unittest.mock._patch) – Patch to start.

Returns MagicMock acting as patched object.

Return type unittest.mock.MagicMock

assert_records_are_equal(*first*, *second*, *msg=None*)
Assert two Document Store records are equal.

Parameters

- **first** – First record to compare.
- **second** – Second record to compare.
- **msg** – Optional message to output on assertion.

assert_records_are_not_equal(*first*, *second*, *msg=None*)
Assert two Document Store records are not equal.

Parameters

- **first** – First record to compare.
- **second** – Second record to compare.
- **msg** – Optional message to output on assertion.

assert_mock_meth_has_calls(*mock_meth*, *call*, **calls*)
Assert moch_meth was called with arguments.

This calls Mock.assert_has_calls and tests for call count. The actual benefit of using this method over the built-in assert_has_calls is that this method tries to pinpoint the actual call that was missing when assert_has_calls raised AssertionError. This is useful when mock_meth has had multiple calls. The built-in assert_has_calls will notify of all calls that the mock_meth has had, while this method will notify of the actual call that was missing.

Parameters

- **mock_meth** – Mocked method that is target of testing.
- **call** – Call that should be found. Instance of `unittest.mock._Call` Repeat this argument to test for multiple calls.

Raises `AssertionError` if calls not found.

class kuha_common.testing.testcases.**KuhaEndToEndTestCase**(*methodName='runTest'*)
Base class for end-to-end tests.

- HTTPClient for interacting with Document Store.
- Assertion methods to check returning payload and status codes.

static load_cli_args(*sysexit_to_skiptest=False*)
Load command line arguments. Setup Document Store URL.

Parameters **sysexit_to_skiptest** (*bool*) – Mask SystemExit as `unittest.SkipTest`. Useful when missing command line arguments should not terminate the test run, but skip tests requiring the arguments.

Returns arguments not known to `kuha_common.cli_setup.settings` (= arguments external to Kuha)

Return type list

static get_record_url(*rec_or_coll*, *_id=None*)
Get URL to Document Store records or single record.

Parameters

- **rec_or_coll** – record, record class or collection
- **_id** (*str or None*) – Optional record ID.

Returns URL to Document Store collection or single record.

Return type str

static get_query_url(*rec_or_coll*, *query_type=None*)
Get URL to Document Store query endpoint for `collection`

Parameters

- **rec_or_coll** (*str, record, or record class*) – Collection to query.

- **query_type** – Optional query type

Returns URL to query endpoint.

Return type str

classmethod GET_to_document_store(*rec_or_coll*, *_id=None*)
GET to Document Store returns record(s).

Parameters

- **rec_or_coll** – record or collection to get.
- **_id** – Optional ObjectId. Will take precedence over `rec_or_coll` id.

Returns response body

classmethod POST_to_document_store(*record*)
POST to Document Store creates record.

Parameters **record** – Record to post.

Returns response body

classmethod DELETE_to_document_store(*rec_or_coll=None*, *_id=None*)
DELETE to Document Store deletes record(s).

Call without arguments to delete all records from all collections.

Parameters

- **rec_or_coll** (*str or None*) – Collection to delete from.
- **_id** (*str or None*) – ID of the record to delete.

Returns None

classmethod query_document_store(*rec_or_coll*, *query*, *query_type=None*)
Execute query against Document Store query API.

Parameters

- **rec_or_coll** (*str or record class or record instance*) – Collection to query.
- **query** – Query.
- **query_type** – Type of Query.

Returns query results

Return type None if query returned no results, dict for results.

classmethod get_collection_record_count(*rec_or_coll*)
Return number or records for collection in Document Store.

Parameters **rec_or_coll** – Document Store record, Document Store record class or collection.

Returns record count in Document Store.

Return type int

assert_document_store_is_empty()
Assert Document Store contains no records.

Raises `AssertionError` if Document Store has records.

3.8.2 kuha_document_store

Kuha Document Store application

Query, manipulate and import Document Store records via HTTP API.

configure.py

Configure Document Store.

kuha_document_store.configure.**add_database_configs**()
Add database configuration values to be parsed.

kuha_document_store.configure.**configure**()
Get settings for application configuration.

Declares application spesific configuration options and some common options declared in *kuha_common.cli_setup*

Configure application with arguments specified in configuration file, environment variables and command line arguments.

Note Calling this function multiple times will not initiate new settings to be parsed, but will return previously parsed settings instead.

Returns settings

Return type argparse.Namespace

serve.py

Main entry point for starting Document Store server.

kuha_document_store.serve.**get_app**(*api_version*, *app_settings=None*)
Setup routes and return initialized Tornado web application.

Parameters

- **api_version** (*str*) – HTTP Api version gets prepended to routes.
- **app_settings** (*dict or None.*) – Settings to store to application.

Returns Tornado web application.

Return type tornado.web.Application

kuha_document_store.serve.**main**()
Application main function.

Parse commandline for settings. Initialize database and web application. Start serving via *kuha_common.server.serve()*. Exit on exceptions propagated at this level.

Returns exit code, 1 on error, 0 on success.

Return type int

handlers.py

Define handlers for responding to HTTP-requests.

class kuha_document_store.handlers.**BaseHandler**(**args*, ***kwargs*)
BaseHandler to derive from.

Provides common methods for subclasses.

Note use from a subclass

prepare()
Prepare for each request.

Set output content type.

get_db()
Get database object stored in settings.

Returns database object.

Return type *kuha_document_store.database.DocumentStoreDatabase*

assert_body_not_empty(*msg=None*)
Assert that request body contains data.

kuha_common.server.BadRequest is raised if body is empty.

Parameters **msg** (*str*) – Optional message for exception.

Raises *kuha_common.server.BadRequest* if body is empty.

class kuha_document_store.handlers.**RestApiHandler**(**args*, ***kwargs*)
Handle requests to REST api.

get(*collection*, *resource_id=None*)
HTTP-GET to REST api endpoint.

Respond with single record or multiple records, depending on whether `resource_id` is requested.

Note Results will be streamed.

Parameters

- **collection** (*str*) – type of the requested collection.
- **resource_id** (*str or None*) – optional ID of the requested resource. If left out of request, will return all records of requested type.

Raises *kuha_common.server.BadRequest* if there are recoverable errors in database operation. The error message is passed to BadRequest. See: *kuha_document_store.database.DocumentStoreDatabase.recoverable_errors*

Raises *kuha_common.server.ResourceNotFound* if requested resource_id does not return results.

post(*collection*, *resource_id=None*)
HTTP-POST to REST api endpoint.

Create new resource from data submitted in request body.

Parameters

- **collection** (*str*) – collection type to create.
- **resource_id** (*str or None*) – receives resource_id for completeness in handler configuration. It is however a *kuha_common.server.BadRequest* if one is submitted.

Raises *kuha_common.server.BadRequest* if request contains resource_id or if database operations raise recoverable errors. See: *kuha_document_store.database.DocumentStoreDatabase.recoverable_errors*

put(*collection*, *resource_id=None*)
HTTP-PUT to REST api endpoint.

Replace existing resource with data in request body.

Parameters

- **collection** (*str*) – collection type to replace.
- **resource_id** (*str or None*) – resource ID to replace. Optional for completeness in handler configuration. It is however a *kuha_common.server.BadRequest* if not submitted.

Raises *kuha_common.server.BadRequest* if requested endpoint does not contain resource_id or if database operation raises one of *kuha_document_store.database.DocumentStoreDatabase.recoverable_errors*

Raises *kuha_common.server.ResourceNotFound* if resource_id returns no results.

delete(*collection*, *resource_id=None*)
HTTP-DELETE to REST api endpoint.

Delete resource or all resources of certain type.

Parameters

- **collection** (*str*) – type of collection
- **resource_id** (*str or None*) – resource ID to delete.

Raises *kuha_common.server.BadRequest* if database operation raises one of *kuha_document_store.database.DocumentStoreDatabase.recoverable_errors*

Raises *kuha_common.server.ResourceNotFound* if resoure_id returns no results.

class kuha_document_store.handlers.**ImportHandler**(**args*, ***kwargs*)
Handle request to import endpoint.

prepare()
Prepare for each request.

All requests must define content type for XML. All requests must contain body data.

post(*importer_id*, *collection=None*)
HTTP-POST to import endpoint.

Lookup correct importer. Load iterative parser. Pass iterative parser to database for processing.

Parameters

- **importer_id** (*str*) – importer to use for importing.
- **collection** (*str or None*) – Optional parameter limits the import to a spesific collection (resource type).

class kuha_document_store.handlers.**QueryHandler**(**args*, ***kwargs*)
Handle request to query endpoint.

Note Results will be streamed.

prepare()
Prepare for each request.

Request content type must be JSON. Request body must not be empty. Requested query type must be supported and query must have valid parameters.

post(*collection*)
HTTP-POST to query endpoint.

Streams the results one JSON document at a time. Thus, the result of a response for multiple records will not a a valid JSON document.

Note Body must be a JSON object.

Parameters **collection** (*str*) – collection (resource type) to query.

database.py

Database module provides access to MongoDB database.

MongoDB Database is accessed throught this module. The module also provides convenience methods for easy access and manipulation via Document Store records defined in *kuha_common.document_store.records*

Database can be used directly, via records or with JSON representation of records.

note This module has strict dependency to *kuha_common.document_store.records*

kuha_document_store.database.**mongodburi**(*host_port*, **hosts_ports*, *database=None*, *credentials=None*, *options=None*)
Create and return a mongodb connection string in the form of a MongoURI.

The standard URI connection scheme has the form: mongodb://[username:password@]host1[:port1][,...hostN[:portN]]][/[database][?options]]

- https://docs.mongodb.com/manual/reference/connection-string/

Parameters

- **host_port** (*str*) – One of more host and port of a mongod instance.
- **database** (*str*) – Optional database.
- **credentials** (*tuple*) – Options credentials (user, pwd).
- **options** (*list*) – Optional options as a list of tuples [(opt_key1, opt_val1), (opt_key2, opt_val2)]

Returns MongoURI connection string.

Return type str

class kuha_document_store.database.**RecordsCollection**(*record_class*, *indexes_unique=None*, *indexes=None*, *validators=None*)
Database collection.

Note Relational Database term *table* is called a *collection* in MongoDB.

Contains properties for Document Store collections. Has strict dependency to *kuha_common.document_store.records*

Parameters

- **record_class** (*kuha_common.document_store.records.Study* or *kuha_common.document_store.records.Variable* or *kuha_common.document_store.records.Question* or *kuha_common.document_store.records.StudyGroup*) – Class of a record that belongs to this collection.
- **indexes_unique** (*list or None*) – declare unique indexes.
- **indexes** (*list or None*) – additional indexes
- **validators** (*list or None*) – additional validators

Returns *RecordsCollection*

isodate_fields = ['_metadata.created', '_metadata.updated']
List common isodate fields

object_id_fields = ['_id']
Fields containing MongoDB ObjectIDs

index_updated = [('_metadata.updated', -1)]
Declare updated field as index.

classmethod bson_to_json(*_dict*)
Encode BSON dictionary to JSON.

Encodes special type of dictionary that comes from MongoDB queries to JSON representation. Also converts datetimes to strings.

Parameters **_dict** (*dict*) – Source object containing BSON.

Returns Source object converted to JSON.

Return type str

get_validator()
Get defined database-level validators.

Note All validators are combined with AND operator.

Returns Database level validators to be used on DB setup.

Return type dict

process_json_for_upsert(*json_document*, *old_metadata=None*)
Preprocess JSON for insert/update operations.

Decodes JSON to Python dictionary. Validates the result. Creates metadata for the document if the document has none, otherwise uses the submitted metadata. Decodes submitted metadata datestamps to datetime objects.

Parameters

- **json_document** (*str*) – JSON representation of a record.
- **old_metadata** (*dict or None*) – old metadata if updating existing record.

Returns Document ready to be submitted to database.

Return type dict

kuha_document_store.database.**RECORD_COLLECTIONS = [<kuha_document_store.database.RecordsCo**
Define Record Collections

class kuha_document_store.database.**Database**(*settings*)
MongoDB database.

Provides access to low-level database operations. For fine access control uses two database credentials, one for read-only operations, one for write operations. Chooses the correct credentials to authenticate based on the operation to be performed.

Note Does not authenticate or connect to the database before actually performing operations that need connecting. Therefore connection/authentication issues will raise when performing operations and not when initiating the database.

Parameters **settings** (argparse.Namespace) – settings for database connections

Returns *Database*

close()
Close open sockets to database.

query_single(*collection_name*, *query*, *fields=None*, *callback=None*)
Query for a single database document.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **query** (*dict*) – Database query.
- **fields** (*list or None*) – Fields to select. None selects all.
- **callback** (*function or None*) – Result callback. Called with result as parameter. If None this method will return the result.

Returns A single document or None if no matching document is found. or if callback is given.

Return type dict or None

query_multiple(*collection_name*, *query*, *callback*, *fields=None*, *skip=0*, *sort_by=None*, *sort_order=1*, *limit=0*)
Query for multiple database documents.

Note has mandatory callback parameter.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **query** (*dict*) – Database query.
- **callback** (*Function that receives single record result as argument.*) – Result callback. Called with each document as parameter.
- **fields** (*list or None*) – Fields to select. None selects all.
- **skip** (*int*) – Skip documents from the beginning of query.
- **sort_by** (*str*) – Sort by field.
- **sort_order** (*int*) – Sort by ascending or descending order. MongoDB users 1 to sort ascending -1 to sort descending.
- **limit** (*int*) – Limit the number of returning documents. 0 returns all documents.

query_distinct(*collection_name*, *fieldname*, *filter_=None*)
Query for distinct values in collection field.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **fieldname** (*str*) – Field to query for distinct values.

- **filter** (*dict or None*) – Optional filter to use with query.

Returns distinct values.

Return type list

count(*collection_name*, *filter_=None*)
Query for document count.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **filter** (*dict or None*) – Optional filter to use for query.

Returns Count of documents.

Return type int

insert(*collection_name*, *document*)
Insert single document to database.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **document** (*dict*) – Document to insert.

Returns Insert result

Return type pymongo.results.InsertOneResult

replace(*collection_name*, *oid*, *document*)
Replace single document in database.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **oid** (*str*) – MongoDB object ID as string.
- **document** (*dict*) – Document to store.

Returns Update result.

Return type pymongo.results.UpdateResult

insert_or_replace(*collection_name*, *query*, *document*)
Insert or replace a single document in database.

Uses special MongoDB method which will replace an existing document if one is found via query. Otherwise it will insert a new document.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **query** (*dict*) – Database query.
- **document** (*dict*) – Document to store.

Returns The document that was stored.

Return type dict

delete_one(*collection_name*, *query*)
Delete single document.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **query** (*dict*) – Database query.

Returns Delete result

Return type pymongo.results.DeleteResult

delete_many(*collection_name*, *query*)
Delete multiple documents.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **query** (*dict*) – Database query.

Returns Delete result

Return type pymongo.results.DeleteResult

class kuha_document_store.database.**DocumentStoreDatabase**(*settings*)
Subclass of *Database*

Provides specialized methods extending the functionality of *Database*. Combines database operations with properties of RecordsCollection. Defines exceptions that, when raised, the HTTP-response operation can continue.

recoverable_errors = (<class 'pymongo.errors.WriteError'>, <class 'json.decoder.JSONDe
These are exceptions that may be raised in normal database operation, so they are not exceptions that should terminate the HTTP-response process. As such, the caller may want to catch these errors.

static json_decode(*json_object*)
Helper method for converting HTTP input JSON to python dictionary.

Parameters **json_object** (*str*) – json to convert.

Returns JSON object converted to python dictionary.

Return type dict

query_multiple(*collection_name*, *query*, *callback*, ***kwargs*)
Query multiple documents with callback.

Converts resulting BSON to JSON. Calls callback with each resulting record JSON.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **query** (*dict*) – Database query.
- **callback** (*function*) – Result callback. Called with each document as parameter.
- ****kwargs** – additional keyword arguments passed to super method.

query_by_oid(*collection_name*, *oid*, *callback*, *fields=None*, *not_found_exception=None*)
Query single record by ObjectID with callback.

Converts BSON result to JSON. Calls the callback with resulting JSON. If parameter for *not_found_exception* is given, will raise the exception if query ObjectID points to no known database object.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **oid** (*str*) – ObjectID to query for.

- **callback** (*function*) – function to call with resulting JSON.
- **fields** (*list or None*) – Fields to select. None selects all.
- **not_found_exception** (*Exception class.*) – Raised if ObjectID not found.

query_distinct(*collection_name*, *fieldname*, *filter_=None*)
Query for distinct values in collection field.

If *fieldname* points to a leaf node, returns a list of values, if it points to a branch node, returns a list of dictionaries.

If *fieldname* points to leaf node of isodate representations, or to branch node that contains isodates, converts datetimes to datestamps which are JSON serializable.

If 'fieldname' points to a leaf node containing MongoDB ObjectID values, cast those values to string.

Note Requires changes to logic if collection.object_id_fields should contain paths with multiple components, for example 'some.path.with.id'. In that case distinct queries that point to brach nodes with OIDs will fail with Exception TypeError: ObjectId('...') is not JSON serializable.

Note Distinction will not work as expected on datestamp-fields that are stored as signed 64-bit integers with millisecond precision. The returned datestamps are not as precise since they have second precision.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **fieldname** (*str*) – Field to query for distinct values.
- **filter** (*dict or None*) – Optional filter to use with query.

Returns distinct values from database

Return type list

insert_or_update_record(*record*)
Insert or update database document by Document Store record.

Special method that takes a Document Store record instance as parameter and determines whether to insert or update the given record.

Makes a query to MongoDB to determine if the record is already in database. If there is a record, calls the record instance's updates_record method to update the instance with values that are present in database but not in the submitted instance.

Afterwards calls `insert_or_replace()` with record instances dictionary representation.

Parameters **record** (*kuha_common.document_store.records.Study* or *kuha_common.document_store.records.Variable* or *kuha_common.document_store.records.Question* or *kuha_common.document_store.records.StudyGroup*) – Document Store record instance.

Returns operation details: {'operation': 'insert'|'update', 'id': <ObjectID>, <records-unique-values>}

Return type dict

bulk_insert_or_update_record(*records*)
Run bulk insert/update operations for Document Store records.

Method that takes an iterable parameter yielding Document Store records. Then calls *insert_or_update_record()* with each record instance.

Parameters **records** (*iterable*) – Document Store records.

Returns list of insert_or_update_record methods operation details.

Return type list

insert_json(*collection_name*, *json_object*)
Insert JSON-encoded document to Database.

Special method that takes a JSON object that is then inserted to database.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **json_object** (*str*) – JSON object representing collection document.

Returns Insert result.

Return type pymongo.results.InsertOneResult

replace_json(*collection_name*, *oid*, *json_object*, *not_found_exception*)
Replace JSON-encoded document in Database.

Special method that replaces a document in database with document given as parameter *json_object*. The document to be replaced is queried by given *oid*.

This method also takes a *not_found_exception* as mandatory parameter. The exception is raised if a document with given *oid* cannot be found.

Note if the submitted JSON does not contain metadata for the document. the metadata gets calculated by *RecordsCollection.process_json_for_upsert()*

Parameters

- **collection_name** (*str*) – Name of database collection.
- **oid** (*str*) – MongoDB object ID as string.
- **json_object** (*str*) – JSON object representing collection document.
- **not_found_exception** (*Exception class.*) – exception to raise if document is not found with *oid*

Returns Update result.

Return type pymongo.results.UpdateResult

delete_by_oid(*collection_name*, *oid*)
Delete database document with ObjectID.

Parameters

- **collection_name** (*str*) – Name of database collection.
- **oid** (*str*) – MongoDB object ID as string.

Returns Delete result

Return type pymongo.results.DeleteResult

validation.py

Simple validation for dictionary representation of document store records.

note This module has strict dependency to *kuha_common.document_store.records*

Validate study record dictionary:

```
>>> from kuha_common.document_store.records import Study
>>> from kuha_document_store.validation import validate
>>> validate(Study.get_collection(), Study().export_dict(include_metadata=False))
Traceback (most recent call last):
[...]
    def validate(collection, document, raise_error=True, update=False):
kuha_document_store.validation.RecordValidationError: ('Validation of studies failed',
    {'study_number': ['null value not allowed']}
)
```

class kuha_document_store.validation.**RecordValidator**(**args*, ***kwargs*)

Subclass `cerberus.Validator` to customize validation.

JSON does not support sets. Therefore a rule to validate list items for uniquity is needed.

For the sake of simplicity in raising and handling validation errors this class also overrides `cerberus.Validator.validate()`.

validate(*document*, ***kwargs*)

Override `cerberus.Validator.validate()`

Handle unvalidated _id-field here to simplify error message flow and enable validation messages.

If document is to be updated it is allowed to have an _id field. If document is being inserted it is an error to have an _id field.

Parameters

- **document** (*dict*) – Document to be validated.
- ****kwargs** – keyword arguments passed to `cerberus.Validator.validate()`. Here it is only checked if keyword argument updated is present and True.

Returns True if validation passes, False if not.

Return type bool

exception kuha_document_store.validation.**RecordValidationError**(*collection*, *validation_errors*, *msg=None*)

Raised on validation errors.

Parameters

- **collection** (*str*) – Collection that got validated.
- **validation_errors** (*dict*) – Validation errors from `cerberus.Validator.errors`. These are stored in `RecordValidationError.validation_errors` for later processing.
- **msg** (*str*) – Optional message.

Returns *RecordValidationError*

class kuha_document_store.validation.**RecordValidationSchema**(*record_class*, **args*)

Create validation schema from records in *kuha_common.document_store.records* to validate user-submitted data.

Schema items are built dynamically by consulting record's field types.

- For single value fields the type is string and null values are not accepted.

- For localizable fields it is required to have a `kuha_common.document_store.constants.REC_FIELDNAME_LANGUAGE` attribute.
- Field attributes are strings and they may be null.
- Subfield values are strings and not nullable.
- Fallback to string, not null.

Record's metadata is accepted as input but not required.

Note `kuha_common.document_store.RecordBase._metadata` and `kuha_common.document_store.RecordBase._id` are also validated at database level.

Seealso *kuha_document_store.database.RecordsCollection.get_validator()*

Every dynamically built schema item may be overriden by a custom schema item given as a parameter for class constructor.

Parameters

- **record_class** (*kuha_common.document_store.records.Study* or *kuha_common.document_store.records.Variable* or *kuha_common.document_store.records.Question* or *kuha_common.document_store.records.StudyGroup*) – class which holds record attributes.
- ***args** – Custom schema items to override dynamically built schema items.

Returns *RecordValidationSchema*

get_schema()
Get Schema.

Returns Validation schema supported by `cerberus`

Return type dict

kuha_document_store.validation.**validate**(*collection*, *document*, *raise_error=True*, *update=False*)
Validate document against collection schema.

Parameters

- **collection** (*str*) – Collection the document belongs to.
- **document** (*dict*) – Document to validate. Document is a dictionary representation of a document store record.
- **raise_error** (*bool*) – Should a *RecordValidationError* be raised if validation fails.
- **update** (*bool*) – Validate for an update/replace operation of an existing record?

Returns True if document passed validation, False if fails.

Return type bool

Raises *RecordValidationError* if raise_error is True and document fails validation.

db_setup.py

Script to help setup Document Store database.

Database administrator may use this script to setup MongoDB instance for usage with Document Store.

kuha_document_store.db_setup.**setup_admin_user**(*admin_username*, *admin_password*, *db*)
Setup administrator credentials.

Note authentication must be disabled in MongoDB to use this operation.

Parameters

- **admin_username** (*str*) – administrator username.
- **admin_password** (*str*) – administrator password.
- **db** (pymongo.database.Database) – MongoDB database

Returns MongoDB database command response

kuha_document_store.db_setup.**setup_users**(*settings*, *client*)
Setup database users for Document Store.

Parameters

- **settings** (argparse.Namespace) – Document Store settings.
- **client** (pymongo.mongo_client.MongoClient) – MongoDB client

Returns list of MongoDB database command responses

kuha_document_store.db_setup.**remove_users**(*settings*, *client*)
Remove Document Store users from database.

Parameters

- **settings** (argparse.Namespace) – Document Store settings.
- **client** (pymongo.mongo_client.MongoClient) – MongoDB client

Returns list of MongoDB database command responses

kuha_document_store.db_setup.**setup_database**(*settings*, *client*)
Create Document Store database.

Parameters

- **settings** (argparse.Namespace) – Document Store settings.
- **client** (pymongo.mongo_client.MongoClient) – MongoDB client

Returns PyMongo Database object

kuha_document_store.db_setup.**delete_database**(*settings*, *client*)
Delete Document Store database.

Parameters

- **settings** (argparse.Namespace) – Document Store settings.
- **client** (pymongo.mongo_client.MongoClient) – MongoDB client

Returns None

kuha_document_store.db_setup.**list_databases**(*settings*, *client*)
List (print) databases.

Note Database won't show in list before it has a collection

Parameters

- **settings** (argparse.Namespace) – Document Store settings.
- **client** (pymongo.mongo_client.MongoClient) – MongoDB client

Returns list of database names

kuha_document_store.db_setup.**setup_collections**(*settings*, *client*)
Setup Document Store collections (tables).

Parameters

- **settings** (argparse.Namespace) – Document Store settings.
- **client** (pymongo.mongo_client.MongoClient) – MongoDB client

Returns list of results

kuha_document_store.db_setup.**delete_collections**(*settings*, *client*)
Delete Document Store collections (tables).

Parameters

- **settings** (argparse.Namespace) – Document Store settings.
- **client** (pymongo.mongo_client.MongoClient) – MongoDB client

Returns list of drop_collection results

kuha_document_store.db_setup.**list_collections**(*settings*, *client*)
List Document Store collections (tables).

Parameters

- **settings** (argparse.Namespace) – Document Store settings.
- **client** (pymongo.mongo_client.MongoClient) – MongoDB client

Returns List of mongodb collections

kuha_document_store.db_setup.**list_db_users**(*settings*, *client*)
List (print) database users.

Parameters

- **settings** (argparse.Namespace) – Document Store settings.
- **client** (pymongo.mongo_client.MongoClient) – MongoDB client

Returns dictionary containing database users and their properties.

kuha_document_store.db_setup.**OPERATIONS = {'remove_users': <function remove_users>, 'setup**
Supported operations.

kuha_document_store.db_setup.**main**()
Script main entry point.

importers

Supported importers are defined in this package.

Declare importers here.

kuha_document_store.importers.**importers = {'ddi_c': <bound method XMLParserBase.from_strin**
Register importers here. {importer_id: importer_function} Importer_id must be unique within importers. Importer_function must accept XML body as string for first argument and Document Store collection as an optional second argument. The importer function must return a generator that will iteratively return populated Document Store record instances.

3.8.3 kuha_oai_pmh_repo_handler

Kuha OAI-PMH Repo Handler application.

Serve records from Kuha Document Store throught OAI-PMH protocol.

serve.py

Main entry point for starting OAI-PMH Repo Handler.

kuha_oai_pmh_repo_handler.serve.**get_app**(*api_version*, *app_settings=None*)
Setup routes and return initialized Tornado web application.

Parameters

- **api_version** (*str*) – HTTP Api version gets prepended to routes.
- **app_settings** (*dict or None.*) – Settings to store to application.

Returns Tornado web application.

Return type tornado.web.Application

kuha_oai_pmh_repo_handler.serve.**main**()
Application main function.

Parse commandline for settings. Setup and serve webapp. Exit on exceptions propagated at this level.

Returns exit code, 1 on error, 0 on success.

Return type int

configure.py

Configure OAI-PMH Repo Handler

kuha_oai_pmh_repo_handler.configure.**configure**(*settings=None*)
Get settings and configure application.

Declares application spesific configuration options and some common options declared in *kuha_common.cli_setup*

Configure application with arguments specified in configuration file, environment variables and command line arguments.

Note Calling this function multiple times will not initiate new settings to be parsed, but will return previously parsed settings instead.

Parameters **settings** – Optional settings to use for configuration. If settings are already loaded this parameter will be silently ignored.

Returns settings

Return type argparse.Namespace

genshi_loader.py

Load genshi templates.

Configure:

```
from genshi_loader import add_template_folder, set_template_writer
add_template_folder(settings.oai_pmh_template_folder)
set_template_writer(handler.template_writer)
```

Use as decorator:

```
from genshi_loader import OAITemplate

class Handler:
...
    @OAITemplate('error.xml')
    async def build_error_message(self):
        return {'msg': 'there was an error'}
```

kuha_oai_pmh_repo_handler.genshi_loader.**FOLDERS = []**
Template folders. There can be multiple.

kuha_oai_pmh_repo_handler.genshi_loader.**add_template_folder**(*folder*)
Add folder to lookup for templates.

Parameters **folder** (*str*) – absolute path to folder containing genshi templates.

kuha_oai_pmh_repo_handler.genshi_loader.**get_template_folder**()
Get template folder.

Returns template folders.

Return type list

kuha_oai_pmh_repo_handler.genshi_loader.**WRITER = []**
Template writer. Function which accepts an iterator as parameter.

kuha_oai_pmh_repo_handler.genshi_loader.**set_template_writer**(*writer*)
Set template writer.

Note Supports only one template writer.

Parameters **writer** – Function that writes the template. Must accept an iterator as a paramter.

kuha_oai_pmh_repo_handler.genshi_loader.**get_template_writer**()
Get template writer.

Returns template writer

Return type function

class kuha_oai_pmh_repo_handler.genshi_loader.**OAITemplate**(*template_file*)
OAITemplate class.

Decorate functions that should write output to genshi-templates. The decorated function must be an asynchronous function and it must return a dictionary.

Example:

```
from genshi_loader import OAITemplate
class Handler:
    @OAITemplate('error.xml')
    async def build_error_message(self):
        ...
        return {'msg': 'there was an error'}
```

Parameters

- **template_file** (*str*) – filename of the template to use.
- **template_folder** (*str*) – optional parameter to use a different template folder to lookup for given template_file.

Raises ValueError if decorated function returns invalid type.

handlers.py

Define handlers for responding to HTTP-requests.

class kuha_oai_pmh_repo_handler.handlers.**OAIRouteHandler**(**args*, ***kwargs*)

Handle requests to OAI endpoint.

OAIRouteHandler extends *kuha_common.server.RequestHandler*.

Input and output goes throught this class. It is responsible for accepting requests via HTTP and routing the requests to OAI-protocol and to the correct verb-handler. Verb-handlers are defined in this class.

Verb-handlers are responsible for calling the *kuha_common.query.QueryController* and again routing the records to OAI-protocol.

Verb-handlers also define the templates used to serialize XML, which is then sent as HTTP-response via *template_writer()*.

The oai protocol is defined in *kuha_oai_pmh_repo_handler.oai.protocol*.

prepare()

Prepare each response.

Initialize response. Load query controller. Set output content type.

template_writer(*generator*)

Writes the output from genshi template.

Parameters **generator** (generator) – generator object containing the XML-serialization.

get()

HTTP-GET handler

Gathers request arguments. Calls router. Finishes the response.

"URLs for GET requests have keyword arguments appended to the base URL"

—http://www.openarchives.org/OAI/openarchivesprotocol.html#ProtocolFeatures

post()

HTTP-POST handler

Validates request content type. Gathers request arguments. Calls router. Finishes the response.

"Keyword arguments are carried in the message body of the HTTP POST. The Content-Type of the request must be application/x-www-form-urlencoded."

—http://www.openarchives.org/OAI/openarchivesprotocol.html#ProtocolFeatures

list_records.py

Run list records sequence on-demand against an OAI-PMH Repo Handler.

Helper script runs through the entire list records sequence with a given metadataPrefix and conditions. Can be used to ensure that all records within a repository are good to serve by catching timeouts from Document Store Client and non-serializable Document Store records.

Logs out the time it takes to complete the full sequence. Prints out all identifiers found by the requested conditions.

If any error conditions are encountered, the best place to look for the cause is the Kuha OAI-PMH Repo Handler log output and Kuha Document Store log output.

exception kuha_oai_pmh_repo_handler.list_records.**InvalidOAIResponse**
The response was not expected.

Raised when:

- HTTP response code is invalid
- Result cannot be parsed as XML
- OAI response has error <error> element

kuha_oai_pmh_repo_handler.list_records.**main**()
Command line interface entry point.

Gather configuration. Setup application. Run sequence and report encountered identifiers.

Returns 0 on success

Return type int

oai

Defines OAI-PMH protocol.

Provides classes for handling requests and responses supported by the protocol. Builds records from *kuha_common.document_store.records*.

oai/errors.py

Errors for OAI-protocol

exception kuha_oai_pmh_repo_handler.oai.errors.**OAIError**(*msg=None*, *context=None*)
Base for OAI errors

get_code()
Get OAI error code

get_msg()
Get OAI error message

get_context()
Get error context

get_contextual_message()
Get error message with possible context.

Returns message with context.

Return type str

exception kuha_oai_pmh_repo_handler.oai.errors.**MissingVerb**(*msg=None*, *context=None*)
OAIError for missing verb

exception kuha_oai_pmh_repo_handler.oai.errors.**BadVerb**(*msg=None*, *context=None*)
OAIError for bad verb

exception kuha_oai_pmh_repo_handler.oai.errors.**NoMetadataFormats**(*msg=None*, *context=None*)
OAIError for no metadata formats

exception kuha_oai_pmh_repo_handler.oai.errors.**IdDoesNotExist**(*msg=None*, *context=None*)
OAIError for no such id

exception kuha_oai_pmh_repo_handler.oai.errors.**BadArgument**(*msg=None*, *context=None*)
OAIError for bad argument

exception kuha_oai_pmh_repo_handler.oai.errors.**CannotDisseminateFormat**(*msg=None*, *context=None*)
OAIError for cannot disseminate format

exception kuha_oai_pmh_repo_handler.oai.errors.**NoRecordsMatch**(*msg=None*, *context=None*)
OAIError for no records match

exception kuha_oai_pmh_repo_handler.oai.errors.**BadResumptionToken**(*msg=None*, *context=None*)
OAIError for bad resumption token

oai/constants.py

OAI constants

kuha_oai_pmh_repo_handler.oai.constants.**REGEX_OAI_IDENTIFIER = "oai:[a-zA-Z][a-zA-Z0-9\\-]+**
Regex to validate oai-identifier. http://www.openarchives.org/OAI/2.0/guidelines-oai-identifier.htm

kuha_oai_pmh_repo_handler.oai.constants.**REGEX_SETSPEC = "([A-Za-z0-9\\-_\\.!~*'\\(\\)])+**
Sets not complying with this regular expression are invalid according to OAI-PMH schema: see: http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd

oai/metadata_formats.py

Define supported metadata formats.

class kuha_oai_pmh_repo_handler.oai.metadata_formats.**MetadataFormatBase**
Base class for metadata formats.

Defines common attributes and methods.

Note This class must be subclassed and the class attributes overriden.

prefix = None
Prefix for metadata format. Override in sublass.

schema = None
Schema URL for metadata format. Override in subclass.

namespace = None
Namespace for metadata format. Override in subclass.

record_fields = None
Record fields. Override in subclass

relative_records = []
Relative records. Override in subclass. Set empty list if no relative records.

get_prefix()
Get metadata prefix.

Returns metadata prefix.

Return type str

get_schema()
Get metadata schema URL.

Returns URL to metadata schema.

Return type str

get_namespace()
Get metadata namespace.

Returns Metadata namespace.

Return type str

get_relative_records()
Get document store records required by this schema.

These fields are required to represent the record in this metadata schema.

Returns list of relative records.

Return type list

get_record_fields(*record=<class 'kuha_common.document_store.records.Study'>*)
Get fields for querying Document Store.

These fields are required to represent the record in this metadata schema.

Parameters **record** (*kuha_common.document_store.records.Study* or *kuha_common.document_store.records.Variable* or *kuha_common.document_store.records.Question* or *kuha_common.document_store.records.StudyGroup*) – Get fields for this Document Store record. Defaults to *kuha_common.document_store.records.Study*

Returns document store record fields

Return type list

Raises KeyError if record is not defined in record_fields

as_dict()
Return metadata attributes in dictionary representation.

Returns metadata attributes.

Return type dict

class kuha_oai_pmh_repo_handler.oai.metadata_formats.**DCMetadataFormat**
Metadata format for OAI-DC.

prefix = 'oai_dc'
Metadata prefix for OAI-DC

schema = 'http://www.openarchives.org/OAI/2.0/oai_dc.xsd'
Metadata schema url for OAI-DC

namespace = 'http://www.openarchives.org/OAI/2.0/oai_dc/'
Namespace for OAI-DC

class kuha_oai_pmh_repo_handler.oai.metadata_formats.**DDIMetadataFormat**
Metadata format for DDI-C.

prefix = 'ddi_c'
Metadata prefix for DDI-C

schema = 'http://www.ddialliance.org/Specification/DDI-Codebook/2.5/XMLSchema/codebook
Metadata schema url for DDI-C

namespace = 'ddi:codebook:2_5'
Namespace for DDI-C

class kuha_oai_pmh_repo_handler.oai.metadata_formats.**CDCDDI25MetadataFormat**
Metadata format for Cessda Data Catalogue DDI 2.5

prefix = 'oai_ddi25'
Metadata prefix for CESSDA Data Catalogue

schema = 'http://www.ddialliance.org/Specification/DDI-Codebook/2.5/XMLSchema/codebook
Metadata schema url for DDI-C

namespace = 'ddi:codebook:2_5'
Namespace for DDI-C

class kuha_oai_pmh_repo_handler.oai.metadata_formats.**EAD3MetadataFormat**
Metadata format for EAD3

oai/protocol.py

Defines the protocol

kuha_oai_pmh_repo_handler.oai.protocol.**as_supported_datetime**(*datetime_str*, *raise_oai_exc=True*)
Convert string representation of datetime to datetime.

Note If the *datetime_str* does not come from HTTP-Request, set *raise_oai_exc* to False.

Note The legitimate formats are YYYY-MM-DD and YYYY-MM-DDThh:mm:ssZ.

Parameters

- **datetime_str** (*str*) – datetime to convert
- **raise_oai_exc** (*bool*) – Catch datetime.strptime errors and reraise as oai-error.

Returns converted datetime.

Return type datetime

Raises kuha_oai_pmh_repo_hander.oai.errors.BadArgument for invalid format if *raise_oai_exc* is True.

kuha_oai_pmh_repo_handler.oai.protocol.**as_supported_datestring**(*datetime_obj*)
Convert datetime to string representation.

The target format is YYYY-MM-DDThh:mm:ssZ

Parameters **datetime_obj** (datetime) – datetime to convert.

Returns string representation of datetime_obj.

Return type str

kuha_oai_pmh_repo_handler.oai.protocol.**encode_uri**(*string*)
Encode uri string.

Replace special characters in string using urllib.parse.quote(). Return resulting string.

Parameters **string** (*str*) – value to encode.

Returns encoded value

Return type str

kuha_oai_pmh_repo_handler.oai.protocol.**decode_uri**(*uri*)
Decode uri string.

Replace uri encoded special characters in string using urllib.parse.unquote(). Return resulting string.

Parameters **string** (*str*) – value to decode.

Returns decoded value

Return type str

kuha_oai_pmh_repo_handler.oai.protocol.**min_increment_step**(*datetime_str*)
Count smallest increment step from datetime string.

Parameters **datetime_str** (*str*) – string representation of a datetime. Datetime must be represented either by day's precision or by second's precision.

Returns smallest increment step.

Rype datetime.timedelta

Raises ValueError if string lenght is invalid.

class kuha_oai_pmh_repo_handler.oai.protocol.**ResumptionToken**(*cursor=0*, *from_=None*, *until=None*, *complete_list_size=None*, *metadata_prefix=None*, *set_=None*)
Class representing OAI-PMH Resumption Token.

Holds attributes of the resumption token. Creates a new resumption token with initial values or takes a dictionary of resumption token arguments. Validates the token based on records list size. If the list size has been changed between requests asserts that the token is invalid by raising a *kuha_oai_pmh_repo_handler.oai.errors.BadResumptionToken* exception.

Note Since OAIArgument.set_ is not supported by resumption token, changing the requested set may result in falsely valid resumption token. But changing the requested set in the middle of a list request sequence should be seen as bad behaviour by the requester/harvester.

Parameters

- **cursor** (*int*) – Optional parameter for the current position in list.
- **from** (*str.*) – Optional parameter for from datestamp. Converted to datetime.datetime on init.
- **until** (*str.*) – Optional parameter for until datestamp. Converted to datetime.datetime on init.

- **complete_list_size** (*int*) – Optional parameter for the umber of records in the complete list.
- **metadata_prefix** (*str*) – Optional parameter for the requested metadata prefix.
- **set** – Optional parameter containing requested set information.

class Attribute(*key*, *value*)
Store ResumptionToken attribute keys and values.

key
Alias for field number 0

value
Alias for field number 1

response_list_size = 100
Configurable value for the size of the list response.

classmethod set_response_list_size(*size*)
Configure response list size.

Parameters **size** (*int*) – Number of records in list response.

classmethod load_resumption_token_argument(*argument*)
Create new resumption token from arguments.

Use to load resumption token from OAI request.

Parameters **argument** (*str*) – Resumption token argument. This comes from HTTP-request.

Returns New *ResumptionToken*

set_complete_list_size(*size*)
Set the number of records in the complete query response.

Note Resumption token is invalid if the number of records for the complete query response has been changed between requests.

Parameters **size** (*int*) – Number of records for the complete query response.

Raises *kuha_oai_pmh_repo_handler.oai.errors.BadResumptionToken* if list sizes don't match.

get_encoded()
Get encoded Resumption Token.

Returns uri-encoded representation of the resumption token if the list request sequence is ongoing. If the list request sequence is over, returns None.

Returns uri-encoded represenation of the token, or None

Return type str or None

class kuha_oai_pmh_repo_handler.oai.protocol.**OAIResponse**(*request_url=None*)
Represents the response.

The response is stored in a dictionary which then gets submitted to XML-templates. Thus it is required that the dictionary built within this class is supported by the templates.

Parameters **request_url** (*str or None*) – Optional requested url. Leave empty to use base url.

classmethod set_repository_name(*name*)
Set repository name.

Parameters **name** (*str*) – repository name.

classmethod set_base_url(*url*)
Set base url

Parameters **url** (*str*) – url.

classmethod set_admin_email(*email*)
Set admin email address.

Parameters **email** (*list*) – Admin email(s)

classmethod set_protocol_version(*version*)
Set protocol version

Parameters **version** (*float*) – OAI-PMH protocol version.

add_record(*record*)
Add record to response

Parameters **record** (*kuha_oai_pmh_repo_handler.oai.records.OAIRecord*) – OAIRecord to add.

has_records()
Return True if response has records.

Return type bool

assert_single_record()
Assert the response has a single record.

Raises AssertionError if there is more or less than a single record.

set_earliest_datestamp(*datestamp*)
Set earliest datestamp.

Parameters **datestamp** (*str*) – datestamp in finest granularity ISO8601

set_deleted_records_declaration(*declaration*)
Set deleted records declaration.

Parameters **declaration** (*str*) – declare support for deleted records

set_granularity(*granularity*)
Set datestamp granularity.

Parameters **granularity** (*str*) – datestamp format for finest granularity supported by this repository.

set_metadata_formats(*metadata_formats*)
Set supported metadata formats.

Parameters **metadata_formats** (*list*) – supported metadata formats

set_resumption_token(*token*)
Set resumption token.

Parameters **token** (*ResumptionToken*) – resumption token.

set_error(*oai_error*)
Set OAI-PMH error.

Note These are the errors that are defined in the OAI-protocol. Programming errors are handled separately in higher levels.

Parameters **oai_error** (Subclass of *kuha_oai_pmh_repo_handler.oai.errors.OAIError*) – OAI error.

add_sets_element(*spec*, *name*)
Add new sets element.

Parameters

- **spec** (*str*) – setSpec-sublement value.
- **name** – setName-sublement value.

extend_sets_element(*sets_list*)
Add multiple sets elements

Note Parameter may come directly from `kuha_oai_pmh_repo_handler.oai.records.Sets.get_sets_list_from_records()`

Parameters **sets_list** (*list*) – list of sets-elements.

set_request_params(*oai_request*)
Gather response parameters from request.

Note These are common response parameters that can be added to each response.

Parameters **oai_request** (*OAIRequest*) – Current OAI-request.

get_response()
Get dictionary representation of the response.

The response attributes are gathered in a dictionary that is to be parsed in the templates.

Note The dictionary will contain python objects, so it is not serializable to JSON or arbitrary formats as is.

Returns Response ready to pass to templates.

Return type dict

class kuha_oai_pmh_repo_handler.oai.protocol.**OAIArguments**(*verb*, *resumption_token=None*, *identifier=None*, *metadata_prefix=None*, *set_=None*, *from_=None*, *until=None*)
Arguments of OAI-protocol.

Store arguments. Convert datestamps string to datetime objects. Validate arguments for each verb.

Parameters

- **verb** (*str*) – requested OAI verb.
- **resumption_token** (*str*) – requested resumption token.
- **identifier** (*str*) – requested identifier.
- **metadata_prefix** (*str*) – requested metadata prefix.
- **set** (*str*) – requested set.
- **from** (*str*) – requested datestamp for from attribute.
- **until** (*str*) – requested datestamp for until attribute.

Raises *kuha_oai_pmh_repo_handler.oai.errors.OAIError* for OAI errors.

supported_verbs = ['Identify', 'ListSets', 'ListMetadataFormats', 'ListIdentifiers', '
Define supported verbs

resumable_verbs = ['ListSets', 'ListIdentifiers', 'ListRecords']
Define resumption token verbs

supported_metadata_formats = [<class 'kuha_oai_pmh_repo_handler.oai.metadata_formats.D
Define supported metadata formats

is_verb_resumable()
Is the requested verb a resumable list request?

Returns True if verb is resumable False otherwise

Return type bool

get_verb()
Get requested OAI-verb.

Returns requested OAI-verb.

Return type str

get_resumption_token()
Get resumption token for request.

The resumption token is either submitted in the request or created automatically.

Returns resumption token.

Return type *ResumptionToken*

get_cursor()
Get resumptionToken cursor

Returns cursor value

Return type str or None

get_from()
Get from argument.

Returns from argument

Return type str or None

get_until()
Get until argument.

Returns until argument.

Return type str or None

get_query_param_until()
Get until datestamp for querying.

Note This is until + smallest increment step.

Returns datestamp of query_param_until attribute.

Return type datetime.datetime

get_identifier()
Get requested identifier.

Returns requested identifier if any.

Return type str or None

get_local_identifier()
Get requested local identifier.

Local identifier does not have prefixes for oai and namespace. It is used to identify records locally.

Returns Local identifier if applicable for the request.

Return type str

Raises *kuha_oai_pmh_repo_handler.oai.errors.IdDoesNotExist* for invalid identifier.

get_metadata_format()
Get requested metadata format.

This is one of the supported metadata formats defined in *OAIArguments.supported_metadata_formats*

Returns requested metadata format if any.

Return type Subclass of *kuha_oai_pmh_repo_handler.oai.metadata_formats.MetadataFormatBase* or None

get_set()
Get requested set.

Returns requested set.

Return type str

is_selective()
Return True if request is selective.

Selective refers to selective harvesting supported by OAI-PMH.

Returns True if selective, False if not.

Return type bool

has_set()
Return True if the request contained set.

Return type bool

iterate_supported_metadata_formats()
Generator for iterating throught supported metadata formats.

Returns Generator object for iterating supported metadata formats.

class kuha_oai_pmh_repo_handler.oai.protocol.**OAIRequest**(*args*)
Represents the OAI request.

Subclass of *OAIArguments*. Defines keys for OAI arguments.

request_attrs = None
Request attributes untouched.

oai/records.py

Define OAI records.

note This module has a strict dependency to *kuha_common.document_store.records*

Contains information for querying records from document store and appending them to responses with *OAIHeaders*, *OAIRecord* and *SETS*.

kuha_oai_pmh_repo_handler.oai.records.**SetAttribute**
Attribute to store set configuration

alias of Set

kuha_oai_pmh_repo_handler.oai.records.**SET_STUDY_GROUP = Set(setname='Study group', setspec=**
Configuration for study_group set

kuha_oai_pmh_repo_handler.oai.records.**SET_LANGUAGE = Set(setname='Language', setspec='langu**
Configuration for language set

kuha_oai_pmh_repo_handler.oai.records.**SET_DATAKIND = Set(setname='Kind of data', setspec='d**
Configuration for datakind set

kuha_oai_pmh_repo_handler.oai.records.**SETS = [Set(setname='Study group', setspec='study_gro**
Supported sets

kuha_oai_pmh_repo_handler.oai.records.**REGEX_VALID_SETSPEC = re.compile("([A-Za-z0-9\\-_\\.**
Validation regex for setspec

kuha_oai_pmh_repo_handler.oai.records.**is_valid_setspec**(*candidate*)
Validates setSpec value.

Parameters **candidate** (*str*) – setSpec value to validate.

Returns True if valid, False if not.

Return type bool

kuha_oai_pmh_repo_handler.oai.records.**get_record_query_field_by_setspec**(*setspec*)
Get document store field to query for set value.

Parameters **setspec** (*str*) – setSpec field of the requested set.

Returns document store field or None

Return type *kuha_common.document_store.field_types.FieldAttribute* or None

kuha_oai_pmh_repo_handler.oai.records.**get_set_specs_from_ds_record**(*ds_record*)
Get set specs from document store record.

Parameters **ds_record** (Record object from *kuha_common.document_store.records*) – One of the document store records. Currently only Study is supported.

Returns set specs for use in oai-headers.

Return type dict

kuha_oai_pmh_repo_handler.oai.records.**get_sets_list_from_query_result**(*set_*, *query_result*)
Get sets list from query results.

Query is built on the basis of set attributes defined in this class. It is a distinct type of query, so the retuned object is not a document store record. This function accepts the results and builds a sets list with each cell containing setName and setSpec keys with their values.

Parameters

- **set** (*SetAttribute*) – set-attribute used for the query.
- **query_result** (*dict*) – results from the query.

Returns list of sets to be used in list sets response.

Return type list

kuha_oai_pmh_repo_handler.oai.records.**get_query_filter_for_set**(*set_request*)
Get filter to use for querying document store.

Returns a dictionary to use for querying document store and filtering by requested set. Returns None if requested set does not exists or is unsupported.

Parameters **set_request** (*str*) – requested set

Returns Query filter or None

Return type dict or None

class kuha_oai_pmh_repo_handler.oai.records.**OAIHeaders**(*identifier*, *datestamp*, ***set_specs*)
Represents OAI-PMH record headers.

Store information of a single record's headers and document store fields to include in query. Provides methods to validate OAI-Identifiers and to iterate set specs list.

Parameters

- **identifier** (*str*) – local identifier of a record.
- **datestamp** (*str*) – last modified/updated datestamp.
- ****set_specs** – key-value pairs of set specs for the record.

namespace_identifier = None
Namespace identifier used to construct an OAI-Identifier Use None if wish to use local identifiers in OAI-responses.

identifier_oai_prefix = 'oai'
Prefix for all identifiers when constructing an OAI-Identifier.

valid_oai_identifier = re.compile("oai:[a-zA-Z][a-zA-Z0-9\\-]*(\\.[a-zA-Z][a-zA-Z0-9
Validation regex for OAI-Identifier

valid_identifier = re.compile("[a-zA-Z0-9\\-_\\.!~*'\\(\\);/\\?:@&=\\+$,%]+")
Validation regex for local identifier (a subset of oai-identifier)

classmethod from_ds_record(*ds_record*)
Return *OAIHeaders* constructed from document store record.

Note Currently supports only Study

Parameters **ds_record** (Record object defined in *kuha_common.document_store.records*) – Document Store record.

Returns headers constructed from Document Store record.

Return type *OAIHeaders*

classmethod set_namespace_identifier(*ns_id*)
Set namespace identifier for all instances.

Note this will be validated afterwards in *set_identifier()*

Parameters **ns_id** (*str*) – namespace identifier

classmethod as_local_id(*identifier*)
Get local identifier part of OAI-Identifier.

Parameters **identifier** (*str*) – records identifier.

Returns local identifier or None for invalid identifier.

Return type str or None

static get_header_fields()
Get header fields to query.

These are the fields required to construct the OAI-HEADER in templates. Check that each OAI-SET field is found here.

Note currently supports only Study.

Returns list of fields to contain in query.

Return type list

set_identifier(*identifier*)
Set identifier.

If namespace_identifier is not None, will build an OAI-Identifier. The identifier will be validated and `ValueError` will be raised if the validation fails.

Parameters **identifier** (*str*) – Record's local identifier.

Raises `ValueError` if validation fails.

get_identifier()
Get identifer

Returns record's identifier.

Return type str

get_datestamp()
Get records datestamp

Returns record's datestamp

Return type str

iterate_set_specs()
Iterate over setSpec key-value pairs.

Returns Generator object for iterating over setSpec key-value pairs.

Return type Generator

class kuha_oai_pmh_repo_handler.oai.records.**OAIRecord**(*study*)
Class stores record and headers.

Parameters **study** (*kuha_common.document_store.records.Study*) – Document Store study record.

add_variable(*variable*)
Add variable to OAIRecord.

Parameters **variable** (*kuha_common.document_store.records.Variable*) – Document Store variable.

add_question(*question*)
Add question to OAIRecord.

Question lookup is done by variable name. Therefore it makes sense to use a dictionary with variable_name as key. The key content will be a list, since a variable may refer multiple questions.

Note questions without variable_name will be discarded and a warning will be logged.

Parameters **question** (*kuha_common.document_store.records.Question*) – Document Store question.

get_questions_by_variable(*variable*)
Get questions for OAIRecord by variable.

Lookup questions by variable's variable_name.

Parameters **variable** (*kuha_common.document_store.records.Variable*) – Document Store variable.

Returns List of *kuha_common.document_store.records.Question*

Return type list

iter_relpubls()
Iterates related publications by distinct description and lang.

Generator yields two-tuples ('lang_desc', 'relpubls'): 'lang_desc' is a two-tuple with first item being the related publication description and the second item being the language of the relpubl element. 'relpubls' is a list containing all bibliographic citation contents of the related publication.

Returns generator that yields tuples (lang_desc, relpubls)

3.8.4 kuha_osmh_repo_handler

Kuha OSMH Repo Handler application.

Serve records from Kuha Document Store throught OSMH protocol.

serve.py

Main entry point for starting OSMH Repo Handler

kuha_osmh_repo_handler.serve.**get_app**(*api_version*, *app_settings=None*)
Setup routes and return initialized Tornado web application.

Parameters

- **api_version** (*str*) – HTTP Api version gets prepended to routes.
- **app_settings** (*dict or None.*) – Settings to store to application.

Returns Tornado web application.

Return type tornado.web.Application

kuha_osmh_repo_handler.serve.**main**()
Application main function.

Parse commandline for settings. Pass settings to kuha_osmh_repo_handler.server.main(). Exit on exceptions propagated at this level.

Returns exit code, 1 on error, 0 on success.

Return type int

configure.py

Configure OSMH Repo Handler

kuha_osmh_repo_handler.configure.**configure**()
Get settings for application configuration.

Declares application spesific configuration options and some common options declared in *kuha_common.cli_setup*

Configure application with arguments specified in configuration file, environment variables and command line arguments.

Note Calling this function multiple times will not initiate new settings to be parsed, but will return previously parsed settings instead.

Returns settings

Return type argparse.Namespace

handlers.py

Define handlers for responding to HTTP-requests.

note OSMH protocol only supports HTTP-GET

class kuha_osmh_repo_handler.handlers.**BaseHandler**(**args*, ***kwargs*)
BaseHandler class for handling OSMH requests.

Derived from *kuha_common.server.RequestHandler*. Defines common attributes.

prepare()
Prepare for responding to request.

Set output content type. Initiate *kuha_osmh_repo_handler.response.RecordsResponse* and *kuha_common.query.QueryController* as instance attributes.

class kuha_osmh_repo_handler.handlers.**ListRecordHeadersHandler**(**args*, ***kwargs*)
Handle list responses.

prepare()
Prepare for each request.

Depending on stream-configuration choose which response callback to use. If streaming is enabled write output once a single record has been built. Otherwise store all records in a list and write output when all records are built.

Note With streaming enabled memory consumption will be lower since the records will not be gathered in a single list and encoded to JSON all at once. When dealing with large repositories the amount of memory consumed without streaming could be an issue.

get(*record_type=None*)
Handles HTTP-GET requests to endpoint.

Parameters **record_type** (*str or None*) – Optional record_type parameter defines if the request limits the list to a certain OSMH type. If the parameter is None, shall output every record in repository. Valid values are defined in handler configuration.

class kuha_osmh_repo_handler.handlers.**GetRecordHandler**(**args*, ***kwargs*)
Handle responses for single record.

get(*record_type*, *identifier*)
Handle HTTP-GET requests to endpoint.

Gathers the needed information by querying Document Store. Builds the OSMH record response.

Parameters

- **record_type** (*str*) – requested OSMH record type.
- **identifier** (*str*) – requested record identifier.

class kuha_osmh_repo_handler.handlers.**ListSupportedRecordTypesHandler**(**args*, ***kwargs*)

Handle response to endpoint that lists supported record types.

get()
Handle HTTP-GET request.

Note returning object will be a list, so will encode to JSON by using build-in json-module.

class kuha_osmh_repo_handler.handlers.**SupportedVersionsHandler**(**args*, ***kwargs*)

Handle response to endpoint that lists supported api version.

Note Currently only single version is supported at a time.

get()
Handle HTTP-GET request.

Note returning object will be a list, so will encode to JSON by using build-in json-module.

response.py

Build responses containing OSMH records.

class kuha_osmh_repo_handler.response.**RecordsResponse**
Response containing records.

Provides methods that can be used as callbacks to *kuha_common.query.QueryController*. Uses OSMH records defined in *kuha_osmh_repo_handler.osmh.records*. Stores these records for later processing.

iterate_records()
Iterate throught records gathered to response.

Returns generator for iterating records.

get_record_appender(*record_constructor*)
Get record appender function.

Records that are appended are constructed by using the record_constructor parameter.

Returns callback function for appending records that get constructed by submitting the receiver parameter to given record_constructor.

Return type functools.partial

get_payload_appender(*record_constructor*)
Get payload appender function.

Records that are appended are constructed by using the record_constructor parameter. After constructing the record, calls the constructed record-objects get_payload() method.

Returns callback function for appending records that get constructed by submitting the receiver parameter to given record_constructor.

Return type functools.partial

get_response()
Build and return the response as a list of dictionary payloads.

Returns record payloads

Return type list

get_single_response()
Get single response payload.

Note After calling this method the payloads list will be empty.

Returns single record's payload.

Return type dict

Raises `ValueError` if contained payloads list has more than a single cell.

osmh

Defines OSMH records and payload.

osmh/records.py

Build OSMH payload from Document Store record objects. Provide mapping between these two record formats. Provide Document Store fields for querying.

note This module has strict dependency to *kuha_common.document_store.records*

class kuha_osmh_repo_handler.osmh.records.**Payload**(*identifier*, *last_modified*)
Represents OSMH record's payload.

Provides methods for manipulating the payload. Stores the payload in a dictionary, which can be easilly encoded to JSON.

Example:

```
>>> from kuha_osmh_repo_handler.osmh.records import Payload
>>> payload = Payload('1', '2017-01-01')
>>> payload.insert_localized_value('study_title', 'en', 'Household Survey')
>>> payload.insert_localized_value('study_title', 'fi', 'Kotitalouskysely')
>>> payload.get() # Indent for better readability
{'identifier': '1',
 'lastModified': '2017-01-01',
 'study_title':
    {'fi': 'Kotitalouskysely',
     'en': 'Household Survey'}
}
```

Parameters

- **identifier** (*str*) – Record's OSMH-identifier. Must uniquelly identify the record within other records of the same OSMH record type in the repository.
- **last_modified** (*str*) – timestamp of the last modification made to the record.

Returns *Payload*

classmethod join_values(**args*)
Join values together using `_join_character`

Parameters ***args** (*str*) – values to join

classmethod split_value(*value*)
Split value using `_join_character`

Parameters **value** (*str*) – value to split

Returns splitted values

Return type list

insert(*key*, *value*)
Insert a value to payload.

Insert a value for given key to the payload. If the key is not present in the payload, creates one.

Parameters

- **key** (*str*) – payload key for the value.
- **value** (*str*) – value to be inserted.

insert_localized_value(*key*, *locale*, *value*)
Insert a localized value to payload.

Insert value for given locale into the given payload key. If the key is not present in the payload, creates one.

Parameters

- **key** (*str*) – payload key
- **locale** (*str*) – values locale
- **value** (*str*) – payload value

append(*key*, *value*, *unique=False*)
Insert list item to given payload key

If key is not in payload, creates it and inserts a list with a single cell containing value. If parameter unique is True, will not append duplicate values to list.

Parameters

- **key** (*str*) – payload key
- **value** (*str*) – value to insert as list item
- **unique** (*bool*) – whether to keep the list of values unique (no duplicates)

header(*osmh_type*)
Create record header to payload

Note Header is common for all record types. The only changing value is the record type.

Parameters **osmh_type** (*str*) – OSMH record type

get()
Return the constructed payload

Returns OSMH payload

Return type dict

class kuha_osmh_repo_handler.osmh.records.**OSMHRecord**(*payload*)
Abstract Base class for OSMH record.

Use from a subclass.

Provides common properties and methods to be used in OSMH records.

Parameters **payload** (*Payload*) – payload of the record.

Raises TypeError if subclass does not define class attributes.

osmh_type
OSMH type. Declare in subclass.

query_document
Document Store record to query. Declare in subclass.

relative_queries_for_record
Does the record-response require relative records queried from Dccument Store. Declare in subclass.

static fields_for_header()
Get fields to query that are required to build the record header.

Override in subclass.

static fields_for_record()
Get fields to query that are required to build the record.

Override in subclass.

static query_filter_for_record(*identifier*)
Get filter which queries the correct record from Document Store.

Override in subclass.

classmethod for_header_response(*ds_record*)
Create a record for response that only contains headers for records.

Parameters **ds_record** (Record defined in *kuha_common.document_store.records*) – Document Store record.

Returns Instantiated OSMH record object.

classmethod for_record_response(*ds_record*)
Create record for response containing the actual record.

Parameters **ds_record** (Record defined in *kuha_common.document_store.records*) – Document Store record.

Returns Instantiated OSHM record object.

classmethod get_query_document()
Return the Document Store record used for Querying.

Returns Document Store record used for querying.

classmethod requires_relative_queries_for_record()
Does the record require querying for relative records from Document Store to construct the full record response.

Returns True or False.

Return type bool

build_header_payload()
Builds the common header payload.

build_record_payload()
Builds the common record payload.

get_payload()
Get the built payload.

Returns record payload.

Return type dict

class kuha_osmh_repo_handler.osmh.records.**StudyRecord**(*study*)
Represents OSMH Study.

Derived from *OSMHRecord*.

Parameters **study** (*kuha_common.document_store.records.Study*) – Study from Document Store.

Returns Instantiated OSMH Study record

Return type *StudyRecord*

query_document
alias of Study

static fields_for_header()
Get fields to query that are required to build the record header.

Returns Study fields required to build record header.

Return type list

static fields_for_record()
Get fields to query that are required to build the record.

Returns Study fields required to build record header.

Return type list

static query_filter_for_record(*identifier*)
Get filter which queries the correct record from Document Store.

Parameters **identifier** (*str*) – study identifier (study number).

Returns filter to use for query.

Return type dict

static get_secondary_query_fields_for_record()
Get fields to query that are required to build the relative record (Variable).

Returns Variable fields.

Return type list

static get_secondary_query_document()
Get secondary query document (Document Store record).

Returns Document Store variable record.

Return type *kuha_common.document_store.records.Variable*

get_secondary_query_filter_for_record()
Get filter which queries the correct record from Document Store.

Returns filter to use for query.

Return type dict

build_relative_record_payload(*relative_record*)
Build payload for relative record.

Parameters **relative_record** (*kuha_common.document_store.records.Variable*) – Relative record instance.

build_record_payload()
Build payload for record.

class kuha_osmh_repo_handler.osmh.records.**VariableRecord**(*variable*)
Represents OSMH Variable.

Derived from *OSMHRecord*.

Parameters **variable** (*kuha_common.document_store.records.Variable*) – Variable from Document Store.

Returns Instantiated OSMH Variable record

Return type *VariableRecord*

query_document
alias of Variable

static fields_for_header()
Get fields to query that are required to build the record header.

Returns Variable fields required to build record header.

Return type list

static fields_for_record()
Get fields to query that are required to build the record.

Returns Variable fields required to build record header.

Return type list

static query_filter_for_record(*identifier*)
Get filter which queries the correct record from Document Store.

Parameters **identifier** (*str*) – variable identifier.

Returns filter to use for query.

Return type dict

build_record_payload()
Build payload for record.

class kuha_osmh_repo_handler.osmh.records.**QuestionRecord**(*question*)
Represents OSMH Question.

Derived from *OSMHRecord*.

Parameters **question** (*kuha_common.document_store.records.Question*) – Question from Document Store.

Returns Instantiated OSMH Question record

Return type *QuestionRecord*

query_document
alias of Question

static fields_for_header()
Get fields to query that are required to build the record header.

Returns Question fields required to build record header.

Return type list

static fields_for_record()
Get fields to query that are required to build the record.

Returns Question fields required to build record header.

Return type list

static query_filter_for_record(*identifier*)
Get filter which queries the correct record from Document Store.

Parameters **identifier** (*str*) – question identifier.

Returns filter to use for query.

Return type dict

build_record_payload()
Build record payload.

class kuha_osmh_repo_handler.osmh.records.**StudyGroupRecord**(*study_group*)
Represents OSMH StudyGroup.

Derived from *OSMHRecord*.

Parameters **study_group** (*kuha_common.document_store.records.StudyGroup*) – StudyGroup from Document Store.

Returns Instantiated OSMH StudyGroup record

Return type *StudyGroupRecord*

query_document
alias of StudyGroup

static fields_for_header()
Get fields to query that are required to build the record header.

Returns StudyGroup fields required to build record header.

Return type list

static fields_for_record()
Get fields to query that are required to build the record.

Returns StudyGroup fields required to build record header.

Return type list

static query_filter_for_record(*identifier*)
Get filter which queries the correct record from Document Store.

Parameters **identifier** (*str*) – Study group identifier.

Returns filter to use for query.

Return type dict

build_record_payload()
Build record payload.

kuha_osmh_repo_handler.osmh.records.**get_osmh_record_for_type**(*osmh_record_type*)
Return the OSMH record class representing *osmh_record_type*.

Parameters **osmh_record_type** (*str*) – Supported OSMH record type.

Returns One of the OSMH records defined in this module.

Return type *StudyRecord* or *VariableRecord* or *QuestionRecord* or *StudyGroupRecord*

3.8.5 kuha_client

kuha_client.py

Kuha Client communicates with Document Store and provides a simple way of importing, updating and deleting records by reading a batch of XML files stored in filesystem.

class kuha_client.kuha_client.**SourceFile**(*path*)
File used as a source for Document Store records.

Parameters **path** – Path to source file

class kuha_client.kuha_client.**FileLog**(*path*)
Keep track of processed files.

Parameters **path** – Path to filelog

set_current(*_file*)
Set current source file.

Parameters **_file** (*SourceFile*) – source file currently processed.

add_pending_study_group(*study_group_identifier*)
Add StudyGroup record to queue waiting to be processed.

Parameters **study_group_identifier** – Id of pending StudyGroup

pop_pending_study_group_files(*study_group_identifier*)
Return and remove source files containing references to `study_group_identifier`.

Parameters **study_group_identifier** – StudyGroup identifier.

Returns source files referencing `study_group_identifier`

Return type list

add_id(*collection*, *_id*)
Add id to current source file's collection of Document Store record IDs.

Parameters

- **collection** (*str*) – Document Store collection the ID belongs to.
- **_id** – ObjectId (ID in Document Store) of the Record.

get_ids(*collection*)
Return list of ObjectIds for `collection` in current file.

Parameters **collection** (*str*) – Document Store collection.

Returns ObjectIds

Return type list

get_filepaths()
Get paths from `self.files`

Iterate throught each *SourceFile* in `self.files` and gather their paths. Return the paths.

Returns List of filepaths

Return type list

load()
Load FileLog from `self.path`. Populates `self.timestamp` and `self.files`.

save()
Save FileLog to `self.path`.

has_match(*path*)
Does the sourcefile found from `path` have a match with path and modified timestamp in this filelog.

Parameters **path** – Path to source file.

Returns True if path and timestamps match.

Return type bool

remove_files_by_path_difference(*paths*)
Remove each *SourceFile* from `self.files` whose path is not in `paths`.

Compare difference in contained source file paths to `paths`. Remove sourcefiles from `self.files` whose paths are not found. Every sourcefile whose paths is not in paths gets removed.

Parameters **paths** – list of filepaths to compare.

exception kuha_client.kuha_client.**DocumentStoreHTTPError**(*error_response*)
Raise if DocumentStore response payload contains errors.

kuha_client.kuha_client.**get_import_url**(*collection=None*, *importer=None*)
Construct URL to Document Store import endpoint.

Parameters

- **collection** (*str*) – Optional parameter to limit the import to certain collection.
- **importer** (*str*) – Optional parameter to set importer. Defaults to 'ddi_c'

Returns Constructed URL

Return type str

kuha_client.kuha_client.**query_record**(*record*)
Query single record by unique fields.

Parameters **record** (*kuha_common.document_store.records.Study* or *kuha_common.document_store.records.Variable* or *kuha_common.document_store.records.Question* or *kuha_common.document_store.records.StudyGroup*) – record to query.

Returns found record if any.

Return type *kuha_common.document_store.records.Study* or *kuha_common.document_store.records.Variable* or *kuha_common.document_store.records.Question* or *kuha_common.document_store.records.StudyGroup*

kuha_client.kuha_client.**query_distinct_ids**(*collection*)
Query collection for distinct ObjectIds

Parameters **collection** (*str*) – record's collection.

Returns set of distinct ids.

Return type *set*

kuha_client.kuha_client.**iterate_xml_directory**(*directory*)
Recursively iterate over XML-files in directory.

Parameters **directory** (*str*) – Absolute path to directory.

Returns generator for iterating XML-files.

kuha_client.kuha_client.**iterate_xml_files_recursively**(**paths*)

Helper for batch processing XML-files.

Check each path. If a path points to a file yield its absolute path. If it points to a directory, recursively iterate paths to each XML-file found and yield each file's absolute path.

Parameters **paths** (*list*) – Paths to file or directory.

Returns generator for iterating absolute paths to xml-files

class kuha_client.kuha_client.**BatchProcessor**(*collections=None*, *file_log=None*, *sourcefiletype=None*)

Processor for operations perfomed in a single run.

Keep record of what gets done. Collect StudyGroups from records and update accordingly. Facilitate access to operations needed to perform tasks against Document Store API.

Parameters

- **collections** (*list or None*) – List of collections to process. Use None to process all collections.
- **file_log** (*FileLog*) – Keep track of processed source files and records ObjectIDs related to them.
- **sourcefiletype** (*str or None*) – Controls how the mapping from sourcefile to Document Store records is done. None sets the default SOURCEFILETYPE_DDIC

classmethod get_supported_sourcefiletypes()

Get supported source file types.

Returns supported source file types.

Return type list

classmethod with_file_log(*file_log_path*, *collections=None*, *sourcefiletype=None*)

Initiate BatchProcessor with File Log.

Parameters

- **file_log_path** (*str*) – path to file log.
- **collections** (*list or None*) – collection to process.
- **sourcefiletype** (*str or None*) – file type of source file.

sourcefileparser(*path*)

Initiate sourcefileparser, which depends on self.sourcefiletype

Parameters **path** – path to source file to be parsed.

Returns iterative parser

create(*record*)

Create new record.

Parameters **record** (*kuha_common.document_store.records.Study* or *kuha_common.document_store.records.Variable* or *kuha_common.document_store.records.Question* or *kuha_common.document_store.records.StudyGroup*) – populated record instance which gets created.

Returns ObjectId of the new record.

Return type str

upsert(*record*)
Upsert record.

If record exists, compare with existing. If records differ, discard the existing record and store the new one to DocumentStore with the existing ObjectId. If record does not exist, insert it to DocumentStore.

Parameters **record** (*kuha_common.document_store.records.Study* or *kuha_common.document_store.records.Variable* or *kuha_common.document_store.records.Question* or *kuha_common.document_store.records.StudyGroup*) – populated record instance which gets created.

Returns ObjectId of the record.

Return type str

upsert_from_parser(*parser*)
Upsert records to `self.collections` from parser.

Parameters **parser** – Parser generates Document Store records.

upsert_paths(**paths*)
Upsert records found recursively from paths.

Parameters ***paths** – one or more paths to recurse to look for files to parse.

upsert_study_groups()
Upsert collected StudyGroups.

add_study_group(*study_group*)
Add StudyGroup for later processing.

Lookup the StudyGroup if it has been stored before and update its contents. If it's not found, store it as a new one.

Parameters **study_group** (*kuha_common.document_store.records.StudyGroup*) – StudyGroup to add for later processing.

import_source(*source_data*)
Import source data to Document Store.

import_source_files(**paths*)
Import files from paths.

Parameters ***paths** – one or more paths to lookup for source files.

remove_absent(*collection*)
Remove records from collection which were not present in current upsert run.

Parameters **collections** (*str*) – collection to process.

remove_absent_records()
Remove records which were not present in current upsert run.

remove_record(*record*)
Remove record or records.

If record is an instance of DocumentStore record, remove it from DocumentStore. If record is a record class, remove all records from it's collection.

Parameters **record** (*DocumentStore record instance or class.*) – Record to remove or class whose records will be removed.

remove_study_and_relatives_by_studyid(*study_id*)
Remove study and relative records.

For a single study the process should remove:

- Actual Study,
- Variable referenced from the Study,
- Questions referenced from the Study,
- Remove references to the Study from StudyGroups.

Note Does not remove StudyGroup even if all references to studies are removed.

Parameters **study_id** (*str*) – ObjectId of the study to remove.

import_run(*lookup_paths*)
Main entry point for import run.

Parameters **lookup_paths** (*list*) – list of paths to lookup for source files.

upsert_run(*lookup_paths*, *remove_absent=False*)
Main entry point for upsert run.

Upsert records found from `lookup_paths`. Remove absent records if *remove_absent* is True.

Parameters

- **lookup_paths** (*sequence*) – list of paths to lookup for source files.
- **remove_absent** (*bool*) – True will remove all records not found from lookup_paths.

remove_run(*record_or_class=None*)
Main entry point for remove run.

Parameters **record_or_class** – Record or RecordClass to remove. If None will remove every record in every collection.

kuha_import.py

Callable module serves as entry point to import records from DocumentStore.

Example run from command line. Import records from /some/path:

```
python -m kuha_client.kuha_import --document-store-url=http://localhost:6001/v0 /some/
↪path
```

Print help:

```
python -m kuha_client.kuha_import -h
```

kuha_client.kuha_import.**import_run**(*paths*, *file_log_path=None*, ***kwargs*)
Import run with arguments.

Parameters

- **paths** – Lookup source files from paths.
- **file_log_path** – Path to file log.
- ****kwargs** – Additional keyword arguments get passed to BatchProcessor.

Returns 0 on success.

Return type int

kuha_client.kuha_import.**cli**()
Parse command line arguments. Call *import_run()*.

Returns Return value of *import_run()*

kuha_upsert.py

Callable module serves as entry point to upsert (insert or update) records from DocumentStore.

Use Document Store's Query API to see if document exists. If it exists, fetch it, update it, submit it back to Document Store via REST API.

Example run from command line. Upsert records from /some/path:

```
python -m kuha_client.kuha_upsert --document-store-url=http://localhost:6001/v0 /some/
↪path
```

Print help:

```
python -m kuha_client.kuha_upsert -h
```

kuha_client.kuha_upsert.**upsert_run**(*paths*, *collections=None*, *file_log_path=None*, *remove_absent=False*, *sourcefiletype=None*)
Upsert run with arguments.

Parameters

- **paths** – Lookup source files from paths.
- **collections** – Limit run to collections.
- **file_log_path** – Path to file log.
- **remove_absent** – Should upsert run remove records, which are found from Document Store but not from source files in current run.
- **sourcefiletype** – File type of source files.

Returns 0 on success.

Return type int

kuha_client.kuha_upsert.**cli**()
Parse command line arguments. Call *upsert_run()*.

Returns Return value of *upsert_run()*

kuha_delete.py

Callable module serves as entry poin to delete records from DocumentStore.

Example run from command line. Delete study with ID:

```
python -m kuha_client.kuha_delete --document-store-url=http://localhost:6001/v0␣
↪studies 5afa741d6fb71d7b2d333982
```

Print help:

```
python -m kuha_client.kuha_delete -h
```

kuha_client.kuha_delete.**cli**()
Parse command line arguments and call BatchProcessor.remove_run()

Returns 0 on success.

Return type int

3.9 Versions

3.9.1 Kuha Common Changelog

0.15.1 (2021-09-06)

- Add missing mapping from DDI 2.5 and DDI 1.2.2 to *Study.study_uris*.

0.15.0 (2021-09-03)

This release ensures future compatibility with CESSDA Data Catalogue.

- Add following attributes to *kuha_common.document_store.records.Study*:
  - document_titles to contain metadata document titles.
  - study_uris to contain URIs linking to study.
- Add DDI mappings to new attributes in *kuha_common.document_store.mappings.ddi*:
  - DDI 2.5 and DDI 1.2.2 use *codeBook/docDscr/citation/titlStmt/titl* to populate *Study.document_titles*
  - DDI 3.1 uses *ddi:DDIInstance/r:Citation/r:Title* to populate *Study.document_titles*
  - DDI 2.5 and DDI 1.2.2 use *codeBook/stdyDscr/citation/holdings/@URI* to populate *Study.study_uris*.
  - DDI 3.1 uses *s:StudyUnit/a:Archive/a:ArchiveSpecific/a:Collection/a:URI* to populate *Study.study_uris*. This XPATH was used to populate *Study.document_uris*, but it seems more appropriate to use it for *Study.study_uris*. There is currently no mapping to populate *Study.document_uris* from DDI 3.1.
- Jenkinsfile uses python-latest to build test environments. In FSD Jenkins python-latest always points to the latest installed python interpreter.
- Add py39 to test environments.
- Remove py35, py36 and py37 from tox test environments.
- Remove tasks running tests with py35, py36 and py37 interpreters from Jenkinsfile.

0.14.0 (2020-06-09)

Python 3.8 compatibility

- *kuha_common.testing.MockCoro* was not working properly with Python 3.8.2 AsyncMock. Switch to a closure function with pruned functionality. Introduce alias *MockCoro* to keep changes more backwards compatible. This is, however, **backwards incompatible change**.
  - MockCoro is now a closure function. Prior 0.14.0 it was a class.
  - MockCoro now returns a coroutine function. Prior 0.14.0 it returned a MockCoro instance.

  - MockCoro func does not get MockCoro instance as the first parameter. Instead the signature can now be exactly same as the mocked out function's. Prior 0.14.0 MockCoro.func was called with MockCoro instance as the first argument.
  - Since mock_coro (alias MockCoro) is no longer a class, it won't hold any attributes. Code relying in mock_coro.func or mock_coro.dummy_rval needs to change.
- Use list(element_copy) instead of deprecated element_copy.getchildren() in *kuha_common.document_store.mappings.xmlbase.element_strip_descendant_text()*
- Add py38 to tox.ini environments. The code should now be Python 3.8. compatible.

Other changes to public APIs

These are backwards compatible changes.

- Add *kuha_common.testing.testcases.KuhaUnitTestCase.assert_mock_meth_has_calls()* to try to pinpoint the exact call that was missing from mocked method.
- Add *kuha_common.cli_setup.add_server_process_count_configuration()* which adds tornado server process count to configuration options.
- Change every *add_** function signature in *kuha_common.cli_setup* to make parser an optional argument. If omitted the functions will use singleton stored in `kuha_common.cli_setup.settings.parser`.

0.13.0 (2020-05-06)

- DDI 2.5 and DDI 1.2.2 mappings use *relpubl/citation/holdings/@URI* and *relpubl/citation/distStmt/distDate* elements that map into Study.related_publications attributes *distribution_date* and *uri*.

0.12.0 (2020-04-28)

- Add `related_publications` to `Study`
- Add DDI 2.5 and DDI 1.2.2 mappings to `Study.related_publications`

0.11.2 (2020-01-24)

- Fix logic in `kuha_common.document_store.field_types.ElementContainer._updates()` where secondary values were compared to other secondary values. This lead to faulty contained values, when a secondary value matched with another secondary value.

0.11.1 (2020-01-09)

- Reset settings in `kuha_common.testing.testcases.load_cli_args()` if tests get skipped, since skipping won't call tearDownClass method

0.11.0 (2020-01-08)

- Initialize argumentparser using configargparse.ArgumentParser directly instead of calling `configargparse.get_arg_parser()` in *kuha_common.cli_setup*. This way the settings get stored only in `kuha_common.cli_setup.settings` singleton.

- Reset settings in `kuha_common.testing.testcases.KuhaEndToEndTestCase.tearDownClass()`.
- Fix faulty call to parent's setUpClass in `kuha_common.testing.testcases.KuhaEndToEndTestCase.tearDownClass()`.

0.10.0 (2019-10-22)

Warning: Breaks backwards compatibility in DDI 1.2.2. and DDI 2.5. mapping of attributes for `Study.study_groups.study_group` and `StudyGroup.study_group_identifier`

Change mappings for DDI 1.2.2. and DDI 2.5.

- Change mappings for Study.study_groups.study_group of DDI 2.5. and DDI 1.2.2. The value for the contained element is now taken from `serStmt/@ID` instead of `serName/@ID`. **Breaks backwards compatibility**
- Change mapping for StudyGroup.study_group_identifier of DDI 2.5. and DDI 1.2.2. The value is now taken from `serStmt/@ID` instead of `serName/@ID`. **Breaks backwards compatibility**

Add new attributes to `kuha_common.document_store.records`

- `Study.study_groups.attr_description`
- `Study.study_groups.attr_uri`
- `Study.data_kinds`
- `Study.data_collection_copyrights`
- `Study.citation_requirements`
- `Study.deposit_requirements`
- `Study.geographic_coverages`
- `StudyGroup.descriptions`
- `StudyGroup.uris`

Add mappings for new attributes

DDI 1.2.2. and DDI 2.5.:

- `/codeBook/stdyDscr/citation/serStmt/serInfo` maps to `Study.study_groups.attr_description`
- `/codeBook/stdyDscr/citation/serStmt/serInfo` maps to `StudyGroups.descriptions`
- `/codeBook/stdyDscr/citation/serStmt/@URI` maps to `Study.study_groups.attr_uri`
- `/codeBook/stdyDscr/citation/serStmt/@URI` maps to `StudyGroup.uris`
- `/codeBook/stdyDscr/citation/prodStmt/copyright` maps to `Study.data_collection_copyrights`
- `/codeBook/stdyDscr/dataAccs/useStmt/citReq` maps to `Study.citation_requirements`

- `/codeBook/stdyDscr/dataAccs/useStmt/deposReq` maps to `Study.deposit_requirements`
- `/codeBook/stdyDscr/stdyInfo/sumDscr/geogCover` maps to `Study.geographic_coverages`
- `/codeBook/stdyDscr/stdyInfo/sumDscr/dataKind` maps to `Study.data_kinds`

DDI 3.1.

- `g:Group/g:Abstract/r:Content` maps to `StudyGroups.descriptions`
- `g:Group/@externalReferenceDefaultURI` maps to `StudyGroups.uris`

0.9.1 (2019-04-03)

- DDI mappings: *kuha_common.document_store.mappings.xmlbase.MappedParams.has_arguments()* treat MappedParams objects which don't have a value and all keyword arguments' values are None as MappedParams objects that don't have any arguments. This way all empty XML elements will be discarded in the mapping phase before they get turned into Document Store reocrds. (Fixes #26)
- Treat attributes-dictionaries with only None values as invalid if the Element's value is also None in *kuha_common.document_store.field_types.Element.add_value()*.
- Servers exception logging *kuha_common.server.log_exception()* will now log all traceback lines, instead of the last four.

0.9.0 (2019-03-14)

- Maintenance release with no new functionality besides bug fixes.
- Support tornado > 5
  - Remove callbacks used with tornado client.
  - requirements.txt declares tornado==6.0.1, which is currently the latest.
- `kuha_common.query.QueryController._query_multiple_with_limit()` now correctly handles `limit` and `skip` query parameters.
- *kuha_common.query.QueryController.query_single()* now correctly raises *kuha_common.document_store.query.QueryException* if passed a `limit` parameter.
- Add validation functions to *kuha_common.document_store.query* to validate query parameters.
- Add common testcase functionality to *kuha_common.testing.testcases.KuhaUnitTestCase*.
- Fix *kuha_common.document_store.client.JSONStreamClient.fetch()* so it won't interfere with queue created by *kuha_common.document_store.client.JSONStreamClient.queue_request()*.
- Update copyright headers to 2019.

0.8.0 (2018-12-18)

- Refactor *kuha_common.server*.
  - *kuha_common.server.serve()* replaces run_server() function. It takes the web-application as a parameter and does not handle application settings. Setting up the application should be handled by the caller who instantiates the application and knows what settings the application needs in order to operate.

- Add *kuha_common.testing.MockCoro* to help mocking out coroutine functions.

0.7.1 (2018-11-20)

- `kuha_common.document_store.mappings.xmlbase.XMLParserBase._get_study_number_from_study_unit_element()` Change priority of elements when looking up Study.study_number:
  1. a:Archive/a:ArchiveSpecific/a:Collection/a:CallNumber
  2. a:Archive/a:ArchiveSpecific/a:Item/a:CallNumber
  3. s:StudyUnit/r:UserID

0.7.0 (2018-11-19)

- **Breaks backward compatibility** Refactored DDI mapping profiles' API. Package structure of *kuha_common.document_store.mappings* changed.
  - *kuha_common.document_store.mappings.xmlbase.XMLParserBase* now mandates how subclasses should be built by providing the public API for callers. The implementation of the actual mapping of resources is an implementation detail of subclasses and does not need to be public.
  - Refactor all DDI mapping profiles to a single module *kuha_common.document_store.mappings.ddi*
  - Add more mapping exceptions to differentiate error conditions in *kuha_common.document_store.mappings.exceptions*
  - Add mapping profile for DDI 3.1. to a new class `kuha_common.document_store.mapping.ddi.DDI31RecordParser` (Implements #24)

0.6.0 (2018-07-18)

- Add methods to *kuha_common.testing.testcases*
- Add support for parsing StudyGroups from DDI 1.2.2. in `kuha_common.document_store.mappings.ddi_122_nesstar.DDI122RecordParser.p_study_groups()` (Implements #20)
- Omit logging of request body to request log if the body is larger than `kuha_common.server.REQUEST_LOG_BODY_MAXLEN`. (Implements #22)
- Fix possible JSONDecodeError in *kuha_common.server.log_request()* (Fixes #23)

0.5.1 (2018-07-11)

- Declare testing.testcases.KuhaUnitTestCase.gen_id() as a classmethod, since it uses class's method.

0.5.0 (2018-07-10)

- Package for common test functions and classes *kuha_common.testing*. (Implements #12)

0.4.1 (2018-07-04)

- `kuha_common.document_store.mappings.xmlbase.XMLMapper._values_from_parent()` resets the state of the attribute mapper if it needs to manipulate the mapper. (Fixes #19)

0.4.0 (2018-07-03)

- Relax record schema by decreasing the number of mandatory attributes. It should be possible to populate an element with attributes but have no value. (Implements #7)
  - *kuha_common.document_store.field_types.Element.add_value()* can be called without a value-parameter.
  - *kuha_common.document_store.field_types.Value.export_dict()* returns an empty dict if value is None.
  - `kuha_common.document_store.field_types.ElementContainer._updates()` If sec_value_value is None and no match is found, create and append new sub-element.
  - `kuha_common.document_store.field_types.ElementContainer._updates()` If sec_value_value is None and a matching value and attributes is found, discard the sec_value completely. (Implements #15)
  - Fix possible KeyError in *kuha_common.document_store.field_types.FieldAttribute.value_from_dict()* This was regression introduced by #7. (Fixes #16)
  - *kuha_common.document_store.field_types.FieldAttribute.value_from_dict()* Now always returns None if nothing was found. Previously it was possible to get an empty list.
- Relax DDI-C mappings: allow record field's value to be None, if there are attributes for the field. (Implements #8)
- Add support for DDI from Nesstar Publisher. (Implements #9)
  - New mapping file ddi_122_nesstar.py to kuha_common.document_store.mappings package.
  - Refactor document_store/mappings/ddi_c.py and separate common XML classes & functions to a new module: xmlbase.py.
    - `kuha_common.document_store.mappings.xmlbase.Value.from_element()` converts empty string values to None.
- Correctly handle mapping from DDI-C if Codebook instance contains multiple series separated by ID. (Fixes #10)
- Fix DDI-C mapping for localized codelists for variables. (Fixes #11)
- DDI-C mapping for Question now removes extra whitepace around research_instruments.
- Fix fetching multiple parent elements for mapped parameters which have no main element. (Fixes #18)

Known Issues

- Mapping is unable to handle DDI-XML (DDI 2.5 and DDI 1.2.2 Nesstar) which contains inconsistent use of conceptual elements. See issue #17. For instance the following structure for anlyUnit:

```
<anlyUnit>Description for analysis unit.
  <concept>concept.of.analysis.unit</concept>
</anlyUnit>
<anlyUnit>Description for another analysis unit.</anlyUnit>
```

Will be interpreted as:

```
[{'analysis_unit': 'concept.of.analysis.unit',
 'description': 'Description for analysis unit.',
 'language': 'en'}]
```

0.3.1 (2018-04-19)

- Add optional keyword arguments to *kuha_common.cli_setup.add_document_store_url()* that are passed to parser.add().
- Declare *kuha_common.cli_setup.Settings.set()* as public function.
- *kuha_common.document_store.field_types.Value.updates()* now actually updates the value.
- Support query filter for record's _id.
- Use document-store-url with full document store base url for *kuha_common.document_store.query.Query*. (Fixes #5)
- Include request body for PUT and PATCH in *kuha_common.server.log_request()*.
- Add *kuha_common.document_store.records.RecordBase.get_id()*.
- Add *kuha_common.document_store.records.RecordBase.updates()*.
- Add updates method for each subclass of *kuha_common.document_store.records.RecordBase*.
- Changes in `kuha_common.document_store.mappings.ddi_c`:
  - Fix default value for universe.attr_clusion.
  - Introduce new class `kuha_common.document_store.mappings.ddi_c.RecordParser` which stores default language; this way parsing can be done simultaneously to multiple XML-documents.
  - Default language is now a mandatory parameter for mapping methods:
    - `kuha_common.document_store.mappings.ddi_c.Value.from_element()`
    - `kuha_common.document_store.mappings.ddi_c.ValueMap.__call__()`
    - `kuha_common.document_store.mappings.ddi_c.MultiValueMap.__call__()`
  - Remove `kuha_common.document_store.mappings.ddi_c.parse()` to streamline the process.
  - Add global to store each parser and corresponding record collection.
  - Declare `kuha_common.document_store.mappings.ddi_c.check_root()` as public function.
  - Add `kuha_common.document_store.mappings.ddi_c.get_root_language()` function.

0.3.0 (2018-03-06)

- Move ddi_c.py mapping module (DDI-C -> Document Store records) from kuha_document_store to kuha_common.document_store.mappings package.
- Forward keyword arguments from Settings.load_parser to configargparse.get_arg_parser in cli_setup.py.
- Make JSONStreamClient._get_request a public method JSONStreamClient.get_request
- Forward keyword arguments from JSONStreamClient.get_request to DocumentStore-Client.streaming_query_request to support more options specifically more HTTP-methods than POST.
- Assert _log_request() in server.py will not raise UnicodeDecodeError if request.body is not utf-8 encoded.
- Add Study.document_uris
- Add abbreviation-attribute to Study.publishers.
- Add DDI-C mappings to Study.document_uris and Study.publishers.attr_abbreviation.

0.2.3 (2018-01-26)

- Implement support for non-localizable containerized elements.
- Add more fields to Study record.
- Add more unit & integration tests.

0.2.2 (2017-11-10)

- Update documentation

0.2.1 (2017-11-09)

- Fix referring variables to questions. Variable may refer multiple questions.
- Fix server.py log_request function. Call RequestHandler.CONTENT_TYPE_JSON, rather than handler-object.
- Partial support for coroutine callbacks in QueryController

0.2.0 (2017-11-01)

- Support referring variables to questions and vice versa.
- Add tox.ini to support running tests with tox.

0.1.0 (2017-10-25)

- Initial release

3.9.2 Kuha Document Store Changelog

0.12.0 (2021-09-06)

- Ensure future compatibility with CESSDA Data Catalogue by adding kuha_common 0.15.1 to requirements.txt. Note that this is not a hard dependency, but must be met in order to follow CDC requirements.
- Add test to ensure the tornado server and mongoclient are started correctly. If motor.motor_tornado.MotorClient is called before starting up tornado server with multiple processes, the program errors out in FileExistsError.
- Drop tests with Python interpreters below py38.

0.11.1 (2021-02-02)

- Lock pip version to 20.3.4 in install script, which is the latest pip that supports Python 3.5. The install script should be compatible with Ubuntu 16.04, which defaults to Python 3.5. The latest pip does not support Python 3.5 and therefore cannot be upgraded.
- Add six==1.15.0 to requirements.txt. It is an indirect dependency required by python-dateutil.
- Add py39 to tox test environments. Use it in Jenkinsfile.
- Switch python3 command to python-latest command in Jenkinsfile. In FSD Jenkins the python-latest always points to the latest installed Python.

0.11.0 (2020-06-12)

- Require kuha_common 0.14.0 or newer in setup.py.
- Support Python 3.8. by upgrading dependencies in requirements.txt:
  - motor==2.1.0
  - pymongo==3.10.1
  - Cerberus==1.3.2
  - kuha_common==0.14.0
- Refactor *kuha_document_store.database* and fit to motor 2.1.0 and pymongo 3.10.1.
- Refactor *kuha_document_store.db_setup* and fit to motor 2.1.0 and pymongo 3.10.1.
- Refactor *kuha_document_store.serve* and *kuha_document_store.configure* and fit to kuha_common 0.14.0.
- Increase test coverage.
- Update documentation regarding supported versions.

0.10.0 (2020-05-07)

- Import DDI 1.2.2. and DDI 2.5. support Study.related_publications
- requirements.txt: kuha_common 0.13.0

0.9.0 (2020-04-29)

- Project source code management changed to git. Does not contain code changes to Document Store.
- INSTALL.rst now instructs to use Git instead of Mercurial.
- requirements.txt: kuha_common 0.12.0

0.8.0 (2020-01-09)

- Kuha Common 0.10.0 introduces new attributes for Study and StudyGroup records.
- Require Kuha Common >= 0.11.0.

0.7.1 (2019-04-05)

- Kuha Common 0.9.1 improves handling of empty XML elements. Use it as a requirement in requirements.txt.
- Add end-to-end test to make sure empty elements are handled correctly when importing.

0.7.0 (2019-03-14)

- Support for kuha_common 0.9.0
- Update copyright headers to 2019.

0.6.0 (2018-12-18)

- Initial support for importing DDI 3.1. metadata.
  - Require kuha_common>=0.8.0

0.5.0 (2018-07-19)

- Support importing StudyGroups from DDI 1.2.2 metadata.
  - Require kuha_common>=0.6.0
- Refactor end-to-end tests: Use `kuha_common.testing` package. (Implements #12)

0.4.2 (2018-07-04)

- Require kuha_common>=0.4.1, because 0.4.0 had critical bug.

0.4.1 (2018-07-04)

- Use tag 0.4.0 for kuha_common in requirements.txt rather than the release branch. This way it is easier to revert to older releases of Document Store.

0.4.0 (2018-07-03)

- Add importer for DDI 1.2.2 Nesstar.

0.3.2 (2018-05-15)

- Relax identifier validation to allow dots.
- Cast MongoDB ObjectId to string for distinct query return values. (Fixes #11)

0.3.1 (2018-03-14)

- Remove extra file kuha_document_store.ini.dist.
- scripts/install_kuha_document_store_virtualenv.sh: install with pip and use –upgrade flag to support upgrading.

0.3.0 (2018-03-06)

- Import API now returns HTTP status code 400, if DataImportError is risen.
- Import API gives result ('import failed') if import has failed, instead of null.
- Fix TypeError on datestamp fields for distinct query type. (Fixes #8)
- Move ddi_c.py mapping module to kuha_common library.
- Fix validation for localizable fields regarding the language attribute. It is now mandatory as it should be. (Fixes #9)
- scripts/install_kuha_document_store_virtualenv.sh: add –upgrade flag to pip install cmd.
- Write some more documentation.

0.2.3 (2018-01-30)

- DDI-C importer: add support for
  - Study.identifiers
  - Study.distributors
  - Study.classifications.attr_uri,
  - Study.classifications.attr_description
  - Study.keywords.attr_uri
  - Study.keywords.attr_description
  - Study.collection_periods
  - Study.analysis_units
  - Study.time_methods
  - Study.sampling_procedures
  - Study.collection_modes
  - Study.data_access_descriptions
- Add validation for Study.persistent_identifiers.
- Separate tests in subfolders *end_to_end*, *integration*, *unit*.
- Add some unit-tests against DDI-C importer. Coverage is still lacking.

0.2.2 (2017-11-10)

- Update documentation.

0.2.1 (2017-11-09)

- **Breaks backward compatibility to 0.2.0. Users should rebuild variable collections.**
- Fix referring variables to questions. Variable may refer multiple questions.

0.2.0 (2017-11-01)

- Support referring variables to questions and vice versa.

0.1.2 (2017-11-01)

- Support querying without created and updated -fields.
- Add tox.ini to support running tests with tox.

0.1.1 (2017-10-27)

- Update documentation.

0.1.0 (2017-10-25)

- Initial version.

3.9.3 Kuha OAI-PMH Repo Handler Changelog

0.14.1 (2021-11-11)

- Metadata rendered by prefix `oai_ddi25` should place distrbtr-element before distDate-element in distStmt. (Fixes #21)

0.14.0 (2021-09-06)

- Ensure future compatibility with CESSDA Data Catalogue
  - Require kuha_common 0.15.1
  - Include elements */codeBook/docDscr/citation/titlStmt/titl* and */codeBook/stdyDscr/citation/holdings/@URI* to DDI 2.5 serializations.
  - Use *study.document_titles* to populate */codeBook/docDscr/citation/titlStmt/titl* and *study.study_uris* to populate */codeBook/stdyDscr/citation/holdings/@URI*.
- Drop testing with Python interpreters below py38.
  - Add py39 to test environments.
  - Remove py35, py36 and py37 from tox test environments.

  - Remove tasks running tests with py35, py36 and py37 interpreters from Jenkinsfile.

0.13.0 (2021-02-02)

- Changes to EAD3 mapping:
  - /ead/archdesc/dsc/c01/c02/did/daoset/dao/@daotype value is now "unknown" instead of "derived"
  - study.principal_investigators are now divided into corpname & persname elements. If a principal investigator has an affiliated organization the value is placed into persname and the organization is placed into corpname. If a principal investigator has no affiliated organization, it's value is expected to be an organization and the value is placed into corpname.
- Lock pip version to 20.3.4 in install script, which is the latest pip that supports Python 3.5. The install script should be compatible with Ubuntu 16.04, which defaults to Python 3.5. The latest pip does not support Python 3.5 and therefore cannot be upgraded into.
- Use ''python-latest" in Jenkinsfile. It is guaranteed to point to the latest python on the CI server. Note that this is a CI specific configuration and is not portable to some arbitrary Jenkins instance.
- Add py39 to tox environments and use it in Jenkinsfile.
- Upgrade to latest genshi 0.7.5 in requirements.txt. Python 3.9 introduced changes to ast module that were incompatible with previous genshi version 0.7.3 used.
- Add six to requirements.txt, since genshi 0.7.5 requires it.

0.12.1 (2020-12-08)

- Fix unhandled AttributeError, when requesting a list response with an unsupported set-parameter. Respond with OAI error code "noRecordsMatch". (Fixes #19)

0.12.0 (2020-06-12)

- Support Python 3.8: involves code changes to tests and requirement of kuha_common 0.14.0. No new functionality or bug fixes are introduced.

0.11.1 (2020-06-05)

- *kuha_oai_pmh_repo_handler.oai.records.get_sets_list_from_query_result()*
  Check for query_result item's set.record_field_setspec using dict's get-method, since the item may not have such key. Remove check for the record_field_setname, since it is acceptable to have an OAI set without a setName. (Fixes #18)

0.11.0 (2020-05-07)

- Use Study.related_publications to render relpubl elements in DDI 2.5. templates.
- require kuha_common 0.13.0

0.10.0 (2020-04-29)

- Project source code management changed to git. Does not contain code changes to OAI-PMH Repo Handler.
- INSTALL.rst now instructs to use Git instead of Mercurial.
- requirements.txt: kuha_common 0.12.0

0.9.0 (2020-03-19)

- Add list_records.py to run through the entire ListRecords sequence on-demand.
- setup.py: Create console script entry point to run list_records.py.
- Add shell script to run the list_records entry point using runtime_env and Python virtualenv.

0.8.0 (2020-01-22)

- Add rights element to OAI-DC serialization.

0.7.1 (2020-01-10)

- Fixes for EAD3 serialization:
  - Correct schemaLocation declaration.
  - Add missing daotype attribute which is required for dao-element.
  - Don't render empty langmaterial elements.

0.7.0 (2020-01-09)

- Support for EAD3 metadata with metadataPrefix `ead3`
- Use study.data_kinds as an OAI-PMH set.
- Render following DDI 2.5. elements in ddi_c and oai_ddi25 metadata:
  - `/codeBook/stdyDscr/citation/prodStmt/copyright`
  - `/codeBook/stdyDscr/dataAccs/useStmt/citReq`
  - `/codeBook/stdyDscr/dataAccs/useStmt/deposReq`
  - `/codeBook/stdyDscr/stdyInfo/sumDscr/geogCover`
  - `/codeBook/stdyDscr/stdyInfo/sumDscr/dataKind`
- Make sure ddi_c and oai_ddi25 templates won't render duplicate ID-attributes per document.
- Add `metadataPrefix` attribute to OAI-Header's request element. (Fixes #15)
- Add `set`, `from` and `until` attributes to OAI-Header's request element.
- Clear OAI-Header's request elements attributes if response results in badVerb or badArgument.
- Relax regex validating OAI-Identifier. (Fixes #16)
- Add set info to resumptionToken and use it in subsequent queries to Document Store. (Fixes #17)
- Fix unhandled exception when requesting a set with more than one colon.

- Update python package requirements: * genshi 0.7.3 * kuha_common 0.10.0

0.6.0 (2019-03-14)

- Support for kuha_common 0.9.0
- Update copyright headers to 2019.

0.5.0 (2018-12-18)

- Require kuha_common 0.8.0. for better support for testing tornado handlers.
- Decouple setup of `template_folder` from importing handlers.

0.4.0 (2018-09-13)

- DDI-C template: Handle the possibility of None values in variable.codelist_codes[n].code. (Fixes #14)
- Support using `base_url` for OAI-PMH request element. (Implements #6)
  - Add configuration option `--oai-pmh-respond-with-requested-url` which defaults to False. Using this computes the base url for OAI-PMH request element from the HTTP request.
- Require kuha_common>=0.6.0.

0.3.0 (2018-07-12)

- Pin requirement to kuha_common version in requirements.txt. This way it is easier to use older releases.
- Fix possible TypeError in *kuha_oai_pmh_repo_handler.oai.records.is_valid_setspec()*, by explicit check for None in *kuha_oai_pmh_repo_handler.oai.records.get_set_specs_from_ds_record()*. (Fixes #11)
- Conceptual container elements should be presented even if no concept is found. (Implements #10)
- Fix possible NameError in `kuha_oai_pmh_repo_handler.oai.OAIRecord.add_question()`. (Fixes #12)
- Serve topcClas and keyword DDI 2.5 elements even if no @ID attribute can be found. (Implements #13)
- Refactor end-to-end tests: Use kuha_common.testing package.

0.2.5 (2018-03-14)

- Add DDI 2.5 metadata format for CESSDA Data Catalogue. (Resolves #8)
- scripts/install_kuha_oai_pmh_repo_handler_virtualenv.sh: Use pip to install & upgrade.

0.2.4 (2018-03-09)

- ListIdentifiers verb handler should not retrieve relative records. (Fixes #7)

0.2.3 (2018-03-07)

- DDI-C metadata & template:
  - Add support for Study.document_uris
  - Add support for Study.publishers.attr_abbreviation
  - Add missing vocabURI-attribute to topClas element
  - Add missing vocabURI-attribute to keyword element
- OAI-DC metadata & template:
  - Use Study.document_uris for identifier.
- Don't require metadataPrefix when request contains resumptionToken. (Fixes #5)
- Add upgrade flag to pip install command.

0.2.2 (2018-01-31)

- DDI-C metadata & template: add support for:
  - Study.identifiers
  - Study.distributors
  - Study.time_methods
  - Study.sampling_procedures
  - Study.collection_modes
  - Study.analysis_units
  - Study.collection_periods
  - Study.data_access_descriptions
- Add support for Study.identifiers in OAI-DC metadata & template.
- Fix incorrect handling of requested identifier when using OAI namespace-identifier. (Fixes #4)
- Separate OAI-PMH error messages for *no known item* and *invalid identifier structure*

0.2.1 (2017-11-16)

- Add ID-attribute to qstn-element in DDI-C template

0.2.0 (2017-11-10)

- Support for variable level metadata in DDI-C

0.1.1 (2017-10-27)

- Update documentation

0.1.0 (2017-10-25)

- Initial release

3.9.4 Kuha OSMH Repo Handler Changelog

0.6.1 (2021-02-02)

- Lock pip version to 20.3.4 in install script, which is the latest pip that supports Python 3.5. The install script should be compatible with Ubuntu 16.04, which defaults to Python 3.5. The latest pip does not support Python 3.5 and therefore cannot be upgraded into.
- Add py39 to tox test environments. Use it in Jenkinsfile.
- Switch python3 command to python-latest command in Jenkinsfile. In FSD Jenkins the python-latest always points to the latest installed Python.

0.6.0 (2020-06-12)

- Support Python 3.8: involves code changes to tests and requirement of kuha_common 0.14.0. No new functionality or bug fixes are introduced.
- Require kuha_common 0.14.0 in requirements.txt and setup.py.

0.5.0 (2020-04-29)

- Project source code management changed to git. Does not contain code changes to OSMH Repo Handler.
- INSTALL.rst now instructs to use Git instead of Mercurial.
- requirements.txt: kuha_common 0.12.0

0.4.0 (2020-01-15)

- requirements.txt: kuha_common 0.11.1. This ensures that OSMH Repo Handler works with the latest Document Store 0.8.0.
- Add dev tools: Jenkinsfile & sonar-project.properties.
- Add more python interpreters to tox.ini.

0.3.0 (2019-03-20)

- requirements.txt: kuha_common 0.9.0, tornado 6.0.1, configargparse 0.14.0.
- handlers.py: Make sure flush_queue callable is set prior calling.
- handlers.py: Fix bug when streaming from empty collection.
- osmh/records.py: Don't write analysis units with empty description to response.

0.2.0 (2018-12-18)

- Require kuha_common 0.8.0 for better support for testing tornado handlers.

0.1.4 (2018-03-14)

- scripts/install_kuha_osmh_repo_handler_virtualenv.sh: use pip to install and upgrade.

0.1.3 (2018-03-07)

- Add analysis unit to study record.
- Use keyword.attr_description for study record subject.
- Add upgrade flag to pip install command.
- Add some tests.
- Change records.Payload._join_character to ':', since it is illegal character in XML attributes. This reduces the risk of possible collissions.

0.1.2 (2017-11-10)

- Update documentation: CHANGES.rst, remove version from kuha_common at README.rst

0.1.1 (2017-10-27)

- Update documentation: configuration.rst, running.rst

0.1.0 (2017-10-25)

- Initial release

3.9.5 Kuha Client Changelog

0.10.0 (2021-09-06)

- Ensure future compatibility with CESSDA Data Catalogue by adding kuha_common 0.15.1 to requirements.txt.
- Drop tests with Python interpreters below py38. Add test environment py39.

0.9.0 (2020-06-12)

- Support Python 3.8
- requirements.txt: kuha_common 0.14.0.

0.8.0 (2020-05-06)

- requirements.txt: kuha_common 0.13.0. Upgrading brings in support for Study.related_publications.

0.7.0 (2020-04-29)

- Project source code management changed to git. Does not contain code changes to Client.
- INSTALL.rst now instructs to use Git instead of Mercurial.
- requirements.txt: kuha_common 0.12.0

0.6.1 (2020-01-24)

- requirements.txt: require kuha_common 0.11.2, which fixes a bug in updating record values.

0.6.0 (2020-01-10)

Warning: Breaks backwards compatibility in DDI 1.2.2. and DDI 2.5. mapping of attributes for `Study.study_groups.study_group` and `StudyGroup.study_group_identifier`. See Kuha Common changelog for more information.

- Kuha Common 0.10.0 introduces new attributes for Study and StudyGroup records.
- Use latest Kuha Common 0.11.1 as a requirement.
- Change end2end test dummydata since Kuha Common 0.10.0 introduces backwards incompatible changes to mapping of `StudyGroup.study_group_identifier`. See Kuha Common changelog for more information.

Note: The running instance of Kuha Document Store must also support Kuha Common >= 0.10.0. for the Client to work. Users should upgrade if needed.

0.5.1 (2019-04-11)

- Kuha Common 0.9.1 improves handling of empty XML elements. Use it as a requirement in requirements.txt.
- Add end-to-end test to ensure empty elements are handled correctly on upsert.
- Call `kuha_common.cli_setup.settings.setup_document_store_query()` in *kuha_client.kuha_delete* to set base url to *kuha_common.document_store.query.Query* (Fixes #10)

0.5.0 (2019-03-20)

- Require kuha_common 0.9.0.
  - Remove `kuha_client.kuha_client.AsyncKuhaClient` since its functionality can now be achieved with *kuha_common.document_store.client.JSONStreamClient*. Refactor HTTP functions to use that instead.
- Make sure Document Store errors are properly logged out. (Fixes #9)
- Update copyright headers to 2019.

0.4.0 (2019-01-17)

- Support for DDI 3.1
- Require kuha_common 0.8.0

0.3.1 (2018-08-30)

- Check that the ObjectId that gets logged is not one that has been deleted before adding the ObjectId to FileLog. (Fixes #8)
- Add debug log messages.

0.3.0 (2018-07-19)

- Pin requirement to kuha_common version in requirements.txt. This way it is easier to use older releases.
- Support for DDI 1.2.2 from Nesstar Publisher. (Implements #7)
  - Require kuha_common>=0.6.0
- Remove files from file log which are not found in current batch. (Fixes #2)

0.2.4 (2018-05-25)

- Fix call for FileLog.add_id(). (Fixes #6)

0.2.3 (2018-05-22)

- Fix regression introduced by 0.2.2. (Fixes #5)

0.2.2 (2018-05-21)

- Fix removing of absent StudyGroups when using file log. (Fixes #4)

0.2.1 (2018-05-16)

- Fix callable module prog paths from help message.

0.2.0 (2018-05-16)

- Complete rewrite of application logic.
- Add tests.
- Support updating and deleting Document Store records.
- Implement file log for keeping track of previously processed files.
- Use clients from kuha_common.client rather than requests-module.
- Update all documentation to match current behaviour.

0.1.2 (2017-11-10)

- Update documentation: CHANGES.rst

0.1.1 (2017-10-27)

- Update documentation: configuration.rst, running.rst

0.1.0 (2017-10-25)

- Initial release

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

k

/(collection)

```
GET /(collection)/(document_id),??
PUT /(collection)/(document_id),??
DELETE /(collection)/(document_id),??
```

/import

```
POST /import/(importer_id)/(collection),
       ??
```

/query

```
POST /query/(collection),??
```

/questions

```
POST /questions,??
```

/studies

```
POST /studies,??
```

/study_groups

```
POST /study_groups,??
```

/variables

```
POST /variables,??
```

Index

Symbols

A

B

C

D

E

F

G

H

K

L

M

N

O

P

Q

R

S

T

U

V

W

X