invenio-oaiserver Documentation

Release 1.0.1

CERN

Dec 17, 2018

Contents

Invenio module that implements an OAI-PMH server.

Further documentation is available on https://invenio-oaiserver.readthedocs.io/

CHAPTER 1

User's Guide

This part of the documentation will show you how to get started in using Invenio-OAIServer.

1.1 Installation

Invenio-OAIServer is on PyPI so all you need is:

```
$ pip install invenio-oaiserver
```

Invenio-OAIServer depends on Invenio-Search, Invenio-Records and Celery/Kombu.

Requirements

Invenio-OAIServer requires a message queue in addition to Elasticsearch (Invenio-Search) and a database (Invenio-Records). See Kombu documentation for list of supported message queues (e.g. RabbitMQ): http://kombu.readthedocs.io/en/latest/introduction.html#transport-comparison

1.2 Configuration

The details of the configuration options for OAI-PMH server.

`invenio_oaiserver.config.`**`OAISERVER_ADMIN_EMAILS = ['info@inveniosoftware.org']`**
The e-mail addresses of administrators of the repository.

It **must** include one or more instances.

`invenio_oaiserver.config.`**`OAISERVER_CACHE_KEY = 'DynamicOAISets::'`**
Key prefix added before all keys in cache server.

`invenio_oaiserver.config.`**`OAISERVER_CELERY_TASK_CHUNK_SIZE = 100`**
Specify the maximum number of records each task will update.

`invenio_oaiserver.config.`**`OAISERVER_CONTROL_NUMBER_FETCHER = 'recid'`**
PIDStore fetcher for the OAI ID control number.

invenio_oaiserver.config.**OAISERVER_DESCRIPTIONS = []**

Specify the optional description containers that can be used to express properties of the repository that are not covered by the standard response to the Identify verb. For further information see: http://www.openarchives.org/OAI/2.0/guidelines.htm

The *eprints*, *oai_identifier* and *friends* description can be added using the helper functions in utils.py as follows:

```
from invenio_oaiserver.utils import eprints_description
from invenio_oaiserver.utils import friends_description
from invenio_oaiserver.utils import oai_identifier_description

OAISERVER_DESCRIPTIONS = [
    eprints_description(metadataPolicy, dataPolicy,
                        submissionPolicy, content),
    oai_identifier_description(scheme, repositoryIdentifier,
                               delimiter, sampleIdentifier),
    friends_description(baseUrls)
]
```

The parameters of each description element are strings if their type is unique or dictionaries, with the type being the key, if it can differ. E.g. the dataPolicy of the eprints element can consist of a text and or URL so it will have the form:

```
metadataPolicy = {'text': 'Metadata can be used by commercial'
                  'and non-commercial service providers',
                  'URL': 'http://arXiv.org/arXiv_metadata_use.htm'}
```

Whereas for the scheme of the oai_identifier it will just be:

```
scheme = 'oai'
```

If the parameter can take an arbitrary amount of elements it can be a list:

```
baseUrls = [http://oai.east.org/foo/,
            http://oai.hq.org/bar/,
            http://oai.south.org/repo.cgi]
```

invenio_oaiserver.config.**OAISERVER_GRANULARITY = 'YYYY-MM-DDThh:mm:ssZ'**

The finest harvesting granularity supported by the repository.

The legitimate values are `YYYY-MM-DD` and `YYYY-MM-DDThh:mm:ssZ` with meanings as defined in ISO8601.

invenio_oaiserver.config.**OAISERVER_METADATA_FORMATS = {'marc21': {'namespace': 'http://w**

Define the metadata formats available from a repository.

Every key represents a `metadataPrefix` and its value has a following structure.

- `schema` - the location of an XML Schema describing the format;
- `namespace` - the namespace of serialized document;
- `serializer` - the importable string or tuple with the importable string and keyword arguments.

Note: If you are migrating an instance running older versions of Invenio<=2.1, you might want to copy settings from `'marc21'` key to `'marcxml'` in order to ensure compatibility for all your OAI-PMH clients.

---

invenio_oaiserver.config.**OAISERVER_PAGE_SIZE = 10**

Define maximum length of list responses.

Request with verbs `ListRecords`, `ListIdentifiers`, and `ListSets` are affected by this option.

invenio_oaiserver.config.**OAISERVER_QUERY_PARSER = 'elasticsearch_dsl:Q'**
Define query parser for OIASet definition.

invenio_oaiserver.config.**OAISERVER_RECORD_INDEX = 'records'**
Specify an Elastic index with records that should be exposed via OAI-PMH.

invenio_oaiserver.config.**OAISERVER_REGISTER_RECORD_SIGNALS = True**
Catch record/set insert/update/delete signals and update the *_oai* field.

invenio_oaiserver.config.**OAISERVER_REGISTER_SET_SIGNALS = True**
Catch set insert/update/delete signals and update the *_oai* record field.

invenio_oaiserver.config.**OAISERVER_RESUMPTION_TOKEN_EXPIRE_TIME = 60**
The expiration time of a resumption token in seconds.

Default: 60 seconds = 1 minute.

Note: Setting longer expiration time may have a negative impact on your Elasticsearch cluster as it might need to keep open cursors.

https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-scroll.html

---

invenio_oaiserver.config.**OAISERVER_XSL_URL = None**
Specify the url (relative or absolute) to the XML Stylesheet file to transform XML OAI 2.0 responses into XHTML.

The url can be a relative path to a local static file:

```
OAISERVER_XSL_URL = '/static/xsl/oai2.xsl'
```

or an absolute url to a remote file (be aware of CORS restrictions):

```
OAISERVER_XSL_URL = 'https//www.otherdomain.org/oai2.xsl'
```

You can obtain an already defined XSL Stylesheet for OAIS 2.0 on EPrints repository (GPLv3 licensed).

1.3 Usage

Invenio module that implements OAI-PMH server.

Invenio-OAIServer is exposing records via OAI-PMH protocol. The core part is reponsible for managing OAI sets that are defined using queries.

OAIServer consists of:

- OAI-PMH 2.0 compatible endpoint.
- Persistent identifier minters, fetchers and providers.
- Backend for formatting Elasticsearch results.

1.3.1 Initialization

Note: You need to have Elasticsearch and a message queue service (e.g. RabbitMQ) running and available on their default ports at 127.0.0.1.

First create a Flask application (Flask-CLI is not needed for Flask version 1.0+):

```
>>> from flask import Flask
>>> app = Flask('myapp')
>>> app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite://'
>>> app.config['CELERY_ALWAYS_EAGER'] = True
>>> if not hasattr(app, 'cli'):
...     from flask_cli import FlaskCLI
...     ext_cli = FlaskCLI(app)
```

There are several dependencies that should be initialized in order to make OAIServer work correctly.

```
>>> from invenio_db import InvenioDB
>>> from invenio_indexer import InvenioIndexer
>>> from invenio_pidstore import InvenioPIDStore
>>> from invenio_records import InvenioRecords
>>> from invenio_search import InvenioSearch
>>> from flask_celeryext import FlaskCeleryExt
>>> ext_db = InvenioDB(app)
>>> ext_indexer = InvenioIndexer(app)
>>> ext_pidstore = InvenioPIDStore(app)
>>> ext_records = InvenioRecords(app)
>>> ext_search = InvenioSearch(app)
>>> ext_celery = FlaskCeleryExt(app)
```

Then you can initialize OAIServer like a normal Flask extension, however you need to set following configuration options first:

```
>>> app.config['OAISERVER_RECORD_INDEX'] = 'marc21',
>>> app.config['OAISERVER_ID_PREFIX'] = 'oai:example:',
>>> from invenio_oaiserver import InvenioOAIServer
>>> ext_oaiserver = InvenioOAIServer(app)
```

Register the Flask Blueprint for OAIServer. If you use InvenioOAIServer as part of the invenio-base setup, the Blueprint will be registered automatically through an entry point.

```
>>> from invenio_oaiserver.views.server import blueprint
>>> app.register_blueprint(blueprint)
```

In order for the following examples to work, you need to work within an Flask application context so let's push one:

```
>>> ctx = app.app_context()
>>> ctx.push()
```

Also, for the examples to work we need to create the database and tables (note, in this example we use an in-memory SQLite database):

```
>>> from invenio_db import db
>>> db.create_all()
```

And create the indices on Elasticsearch.

```
>>> indices = list(ext_search.create(ignore=[400]))
>>> ext_search.flush_and_refresh('_all')
```

1.3.2 Creating OAI sets

"A set is an optional construct for grouping records for the purpose of selective harvesting" *[OAISet]*. The easiest way to create new OAI set is using database model.

```
>>> from invenio_oaiserver.models import OAISet
>>> oaiset = OAISet(spec='higgs', name='Higgs', description='...')
>>> oaiset.search_pattern = 'title:higgs'
>>> db.session.add(oaiset)
>>> db.session.commit()
```

The above set will group all records that contain word "higgs" in the title.

We can now see the set by using verb `ListSets`:

```
>>> with app.test_client() as client:
...     res = client.get('/oai2d?verb=ListSets')
>>> res.status_code
200
>>> b'Higgs' in res.data
True
```

1.3.3 Data model

Response serializer, indexer and search expect `_oai` key in record data with following structure.

```
{
    "_oai": {
        "id": "oai:example:1",
        "sets": ["higgs", "demo"],
        "updated": "2012-07-04T15:00:00Z"
    }
}
```

There **must** exist an `id` key with a non-null value otherwise the record is not exposed via OAI-PHM interface (`listIdentifiers`, `listRecords`). The value of this field should be regitered in PID store. We provide default *`oaiid_minter()`* that can register existing value or mint new one by concatenating a configuration option `OAISERVER_ID_PREFIX` and record value from `control_number` field.

All values in `sets` must exist in `spec` column in `oaiserver_set` table or they will be removed when record updater is executed. The last field `updated` contains ISO8601 datetime of the last record metadata modification according to following rules for selective harvesting.

1.3.4 XSL Stylesheet

OAI 2.0 results can be nicely presented to the user navigating to the OAI Server by defining an XSL Stylesheet to transform XML into HTML.

You can configure the module to use a static XSL file or to fetch it from a remote server.

To use a local XSL Stylesheet, place the file in a *static* folder, and set the relative url in the config *OAISERVER_XSL_URL*. For example:

```
OAISERVER_XSL_URL = '/static/xsl/oai2.xsl'
```

To use a remote XSL Stylesheet, set the config variable to an absolute url:

```
OAISERVER_XSL_URL = 'https://www.mydomain.com/oai2.xsl'
```

Be aware of CORS restrictions when fetching content from remote servers.

You can obtain an already defined XSL Stylesheet for OAIS 2.0 on EPrints repository (GPLv3 licensed).

1.4 Example application

Run ElasticSearch and RabbitMQ servers.

Run the development server:

```
$ pip install -e .[all]
$ cd examples
$ ./app-setup.sh
$ ./app-fixtures.sh
```

Run the server:

```
$ FLASK_APP=app.py flask run --debugger -p 5000
```

Visit http://localhost:5000/admin/oaiset to see the admin interface.

To be able to uninstall the example app:

```
$ ./app-teardown.sh
```

CHAPTER 2

API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

2.1 API Docs

Invenio-OAIServer extension implementation.

class invenio_oaiserver.ext.**InvenioOAIServer**(*app=None*, ***kwargs*)
Invenio-OAIServer extension.

Extension initialization.

Parameters **app** – An instance of `flask.Flask`. (Default: `None`)

init_app(*app*, ***kwargs*)
Flask application initialization.

Parameters **app** – An instance of `flask.Flask`.

init_config(*app*)
Initialize configuration.

Parameters **app** – An instance of `flask.Flask`.

2.1.1 Models

Models for storing information about OAIServer state.

class invenio_oaiserver.models.**OAISet**(***kwargs*)
Information about OAI set.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

add_record(*record*)
Add a record to the OAISet.

Parameters **record** (*invenio_records.api.Record* or derivative.) – Record to be added.

description
Human readable description.

has_record(*record*)
Check if the record blongs to the OAISet.

Parameters **record** (*invenio_records.api.Record* or derivative.) – Record to be checked.

name
Human readable name of the set.

remove_record(*record*)
Remove a record from the OAISet.

Parameters **record** (*invenio_records.api.Record* or derivative.) – Record to be removed.

search_pattern
Search pattern to get records.

spec
Set identifier.

validate_spec(*key*, *value*)
Forbit updates of set identifier.

2.1.2 Views

Views init.

OAI-PMH verbs.

class invenio_oaiserver.verbs.**DateTime**(*format=None*, ***kwargs*)
DateTime with a permissive deserializer.

from_iso_permissive(*use_dateutil=True*)
Parse an ISO8601-formatted datetime and return a datetime object.

Inspired by the marshmallow.utils.from_iso function, but also accepts datestrings that don't contain the time.

class invenio_oaiserver.verbs.**OAISchema**(*extra=None*, *only=None*, *exclude=()*, *prefix=u''*, *strict=None*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*)
Base OAI argument schema.

class Meta
Schema configuration.

validate(*data*)
Check range between dates under keys `from_` and `until`.

class invenio_oaiserver.verbs.**ResumptionVerbs**
List valid verbs when resumption token is defined.

class ListIdentifiers(*extra=None*, *only=None*, *exclude=()*, *prefix=u”*, *strict=None*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*)
Arguments for ListIdentifiers verb.

class ListRecords(*extra=None*, *only=None*, *exclude=()*, *prefix=u”*, *strict=None*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*)
Arguments for ListRecords verb.

class ListSets(*extra=None*, *only=None*, *exclude=()*, *prefix=u”*, *strict=None*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*)
Arguments for ListSets verb.

class invenio_oaiserver.verbs.**Verbs**
List valid verbs and its arguments.

class GetMetadata(*extra=None*, *only=None*, *exclude=()*, *prefix=u”*, *strict=None*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*)
Arguments for GetMetadata verb.

class GetRecord(*extra=None*, *only=None*, *exclude=()*, *prefix=u”*, *strict=None*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*)
Arguments for GetRecord verb.

class Identify(*extra=None*, *only=None*, *exclude=()*, *prefix=u”*, *strict=None*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*)
Arguments for Identify verb.

class ListIdentifiers(*extra=None*, *only=None*, *exclude=()*, *prefix=u”*, *strict=None*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*)
Arguments for ListIdentifiers verb.

class ListMetadataFormats(*extra=None*, *only=None*, *exclude=()*, *prefix=u”*, *strict=None*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*)
Arguments for ListMetadataFormats verb.

class ListRecords(*extra=None*, *only=None*, *exclude=()*, *prefix=u”*, *strict=None*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*)
Arguments for ListRecords verb.

class ListSets(*extra=None*, *only=None*, *exclude=()*, *prefix=u”*, *strict=None*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*)
Arguments for ListSets verb.

invenio_oaiserver.verbs.**make_request_validator**(*request*)
Validate arguments in incomming request.

invenio_oaiserver.verbs.**validate_metadata_prefix**(*value*)
Check metadataPrefix.

Parameters **value** – One of the metadata identifiers configured in OAISERVER_METADATA_FORMATS.

2.1.3 Response

Implement funtions for managing OAI-PMH resumption token.

class invenio_oaiserver.resumption_token.**ResumptionToken**(*default=<marshmallow.missing>*, *attribute=None*, *load_from=None*, *dump_to=None*, *error=None*, *validate=None*, *required=False*, *allow_none=None*, *load_only=False*, *dump_only=False*, *missing=<marshmallow.missing>*, *error_messages=None*, ***metadata*)

Resumption token validator.

class invenio_oaiserver.resumption_token.**ResumptionTokenSchema**(*extra=None*, *only=None*, *exclude=()*, *prefix=u''*, *strict=None*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*)

Schema with resumption token.

load(*data*, *many=None*, *partial=None*)

Deserialize a data structure to an object.

invenio_oaiserver.resumption_token.**serialize**(*pagination*, ***kwargs*)

Return resumption token serializer.

2.1.4 Persistent identifier

OAI-PMH ID provider.

class invenio_oaiserver.provider.**OAIIDProvider**(*pid*)

OAI-PMH identifier provider.

Initialize provider using persistent identifier.

Parameters **pid** – A invenio_pidstore.models.PersistentIdentifier instance.

classmethod create(*object_type=None*, *object_uuid=None*, ***kwargs*)

Create a new record identifier.

Parameters

- **object_type** – The object type. (Default: None)
- **object_uuid** – The object UUID. (Default: None)

default_status = 'K'

OAI IDs are by default registered when object is known.

pid_provider = 'oai'

Provider name.

pid_type = 'oai'
Type of persistent identifier.

Persistent identifier minters.

invenio_oaiserver.minters.**oaiid_minter**(*record_uuid*, *data*)
Mint record identifiers.

Parameters

- **record_uuid** – The record UUID.
- **data** – The record data.

Returns A invenio_pidstore.models.PersistentIdentifier instance.

Persistent identifier fetchers.

invenio_oaiserver.fetchers.**oaiid_fetcher**(*record_uuid*, *data*)
Fetch a record's identifier.

Parameters

- **record_uuid** – The record UUID.
- **data** – The record data.

Returns A invenio_pidstore.fetchers.FetchedPID instance.

2.1.5 Utilities

Utilities.

invenio_oaiserver.utils.**datetime_to_datestamp**(*dt*, *day_granularity=False*)
Transform datetime to datestamp.

Parameters

- **dt** – The datetime to convert.
- **day_granularity** – Set day granularity on datestamp.

Returns The datestamp.

invenio_oaiserver.utils.**dumps_etree**(*pid*, *record*, ***kwargs*)
Dump MARC21 compatible record.

Parameters

- **pid** – The invenio_pidstore.models.PersistentIdentifier instance.
- **record** – The invenio_records.api.Record instance.

Returns A LXML Element instance.

invenio_oaiserver.utils.**eprints_description**(*metadataPolicy*, *dataPolicy*, *submissionPolicy=None*, *content=None*)
Generate the eprints element for the identify response.

The eprints container is used by the e-print community to describe the content and policies of repositories. For the full specification and schema definition visit: http://www.openarchives.org/OAI/2.0/guidelines-eprints.htm

invenio_oaiserver.utils.**friends_description**(*baseURLs*)
Generate the friends element for the identify response.

The friends container is recommended for use by repositories to list confederate repositories. For the schema definition visit: http://www.openarchives.org/OAI/2.0/guidelines-friends.htm

invenio_oaiserver.utils.**oai_identifier_description**(*scheme*, *repositoryIdentifier*, *delimiter*, *sampleIdentifier*)

Generate the oai-identifier element for the identify response.

The OAI identifier format is intended to provide persistent resource identifiers for items in repositories that implement OAI-PMH. For the full specification and schema definition visit: http://www.openarchives.org/OAI/2.0/guidelines-oai-identifier.htm

invenio_oaiserver.utils.**sanitize_unicode**(*value*)

Removes characters incompatible with XML1.0.

Following W3C recommandation : https://www.w3.org/TR/REC-xml/#charsets Based on https://lsimons.wordpress.com/2011/03/17/stripping-illegal-characters-out-of-xml-in-python/ # noqa

invenio_oaiserver.utils.**serializer**(**args*, ***kwds*)

Return etree_dumper instances.

Parameters **metadata_prefix** – One of the metadata identifiers configured in OAISERVER_METADATA_FORMATS.

Error.

exception invenio_oaiserver.errors.**OAIBadMetadataFormatError**

Metadata format required doesn't exist.

exception invenio_oaiserver.errors.**OAISetSpecUpdateError**

Spec attribute cannot be updated.

The correct way is to delete the set and create a new one.

CHAPTER 3

Additional Notes

Notes on how to contribute, legal information and changes are here for the interested.

3.1 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

3.1.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/inveniosoftware/invenio-oaiserver/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

Invenio-OAIServer could always use more documentation, whether as part of the official Invenio-OAIServer docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/inveniosoftware/invenio-oaiserver/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.1.2 Get Started!

Ready to contribute? Here's how to set up *invenio-oaiserver* for local development.

1. Fork the *inveniosoftware/invenio-oaiserver* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/invenio-oaiserver.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv invenio-oaiserver
$ cd invenio-oaiserver/
$ pip install -e .[all]
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass tests:

```
$ ./run-tests.sh
```

   The tests will provide you with test coverage and also check PEP8 (code style), PEP257 (documentation), flake8 as well as build the Sphinx documentation and run doctests.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -s
    -m "component: title without verbs"
    -m "* NEW Adds your new feature."
    -m "* FIX Fixes an existing issue."
    -m "* BETTER Improves and existing feature."
    -m "* Changes something that should not be visible in release notes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

3.1.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests and must not decrease test coverage.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring.
3. The pull request should work for Python 2.7, 3.3, 3.4 and 3.5. Check https://travis-ci.org/inveniosoftware/invenio-oaiserver/pull_requests and make sure that the tests pass for all supported Python versions.

3.2 Changes

Version 1.0.1 (released 2018-12-14)

Version 1.0.0 (released 2018-03-23)

- Initial public release.

3.3 License

3.4 Contributors

- Alexander Ioannidis
- Alizee Pace
- Andrew McCracken
- Diego Rodriguez

- Dinos Kousidis
- Emanuel Dima
- Harri Hirvonsalo
- Harris Tzovanakis
- Jan Aage Lavik
- Jiri Kuncar
- Krzysztof Nowak
- Lars Holm Nielsen
- Leonardo Rossi
- Nicola Tarocco
- Nicolas Harraudeau
- Samuele Kaplun
- Sebastian Witowski
- Tibor Simko
- Wojciech Ziółek

Bibliography

[OAISet] https://www.openarchives.org/OAI/openarchivesprotocol.html#Set

Python Module Index

i

Index

P

R

S

V