
invenio-records-files Documentation

Release 1.2.1

CERN

Jun 16, 2020

Contents

1	User's Guide	3
1.1	Installation	3
1.2	Configuration	3
1.3	Usage	4
2	API Reference	9
2.1	API Docs	9
3	Additional Notes	11
3.1	Contributing	11
3.2	Changes	13
3.3	License	13
3.4	Contributors	14
	Python Module Index	15
	Index	17

Invenio module that provides basic API for integrating [Invenio-Records](#) and [Invenio-Files-REST](#).

Features:

- Records creation
- Files creation
- Accessing files
- Files metadata management
- Files extraction from records

Further documentation is available on <https://invenio-records-files.readthedocs.io/en/latest/usage.html#initializations>

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

1.1 Installation

Invenio-Records-Files is on PyPI so all you need is:

```
$ pip install invenio-records-files
```

1.2 Configuration

Invenio-Records-Files configuration.

```
invenio_records_files.config.RECORDS_FILES_REST_ENDPOINTS = {}
```

REST endpoints configuration.

You can configure the REST API endpoint to access the record's files as follows:

```
RECORDS_FILES_REST_ENDPOINTS = {
    '<*_REST_ENDPOINTS>': {
        '<endpoint-prefix>': '<endpoint-suffix>',
    }
}
```

- `<*_REST_ENDPOINTS>` corresponds to [Invenio-Records-REST endpoint configurations names](#) that you have defined in your application.
- `<endpoint-prefix>` is the unique name of the endpoint configuration as it is defined in [Invenio-Records-REST](#) like configuration. This needs to match an already existing endpoint name in the `<*_REST_ENDPOINTS>` configuration.
- `<endpoint-suffix>` is the endpoint path name to access the record's files.

```
{'recid': '/myawesomefiles'} -> /records/1/myawesomefiles
```

An example of this configuration is provided in the [Integration with Invenio REST API](#) section of the documentation.

1.3 Usage

Integration of records and files for Invenio.

Invenio-Records-Files provides basic API for integrating [Invenio-Records](#) and [Invenio-Files-REST](#).

1.3.1 Initialization

First create a Flask application:

```
>>> from flask import Flask
>>> app = Flask('myapp')
>>> app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite://'
```

Records-Files has no Flask extension, however it is dependent on [Invenio-Records](#) and [Invenio-Files-REST](#) which must be initialized first:

```
>>> from invenio_db import InvenioDB
>>> ext_db = InvenioDB(app)
>>> from invenio_records import InvenioRecords
>>> from invenio_files_rest import InvenioFilesREST
>>> ext_filesrest = InvenioFilesREST(app)
>>> ext_records = InvenioRecords(app)
```

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

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

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

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

Lastly, since you're managing files, you need to create a default location. Here you will create a location in a temporary directory:

```
>>> import tempfile
>>> tmppath = tempfile.mkdtemp()
>>> from invenio_files_rest.models import Location
>>> db.session.add(Location(name='default', uri=tmppath, default=True))
>>> db.session.commit()
```

1.3.2 Creating a record

Import Invenio-Records-Files basic API `invenio_records_files.api.Record`:


```
>>> from invenio_records_files.api import Record
```

This Record class has a special property `files` through which you can access and create files. By default the class creates a bucket when you create a record:

```
>>> record = Record.create({})
>>> len(record.files)
0
```

You can also just create a record without an associated bucket:

```
>>> record_nobucket = Record.create({}, with_bucket=False)
>>> record_nobucket.files is None
True
```

1.3.3 Creating files

You are now ready to create your first file using the Invenio-Records-Files API:

```
>>> from six import BytesIO
>>> record.files['hello.txt'] = BytesIO(b'Hello, World')
```

In the above example you created a file named `hello.txt` as a new object in the record bucket.

1.3.4 Accessing files

You can access the above file through the same API:

```
>>> len(record.files)
1
>>> 'hello.txt' in record.files
True
>>> fileobj = record.files['hello.txt']
>>> print(fileobj.key)
hello.txt
```

1.3.5 Metadata for files

Besides creating files you can also assign metadata to files:

```
>>> fileobj['filetype'] = 'txt'
>>> print(record.files['hello.txt']['filetype'])
txt
```

Certain key names are however reserved:

```
>>> fileobj['key'] = 'test'
Traceback (most recent call last):
...
KeyError: 'key'
```

The reserved key names are all the properties which already exist in `invenio_files_rest.models.ObjectVersion`.

You can however still use the reserved keys for getting metadata:

```
>>> print(fileobj['key'])
hello.txt
```

1.3.6 Dumping files

You can make a dictionary of all files:

```
>>> dump = record.files.dumps()
>>> for k in sorted(dump[0].keys()):
...     print(k)
bucket
checksum
file_id
filetype
key
size
version_id
```

1.3.7 Retrieve files from a record

Invenio-Records-Files provides an utility to retrieve files of a given record.

```
>>> from invenio_records_files.utils import record_file_factory
>>> fileobj = record_file_factory(None, record, 'hello.txt')
>>> print(fileobj.key)
hello.txt
```

If the file does not exist or the record class has no files property, the factory will return None:

```
>>> fileobj = record_file_factory(None, record, 'invalid')
>>> fileobj is None
True
```

Some other Invenio modules such as [Invenio-Previewer](#) already uses it to programmatically access record's files.

1.3.8 Integration with Invenio REST API

Invenio-Records-Files provides REST endpoints to retrieve or upload the files of a record:

```
# Upload a file named example.txt to the record with pid of 1
$ curl -X PUT http://localhost:5000/api/records/1/files/example.txt \
  -H "Content-Type: application/octet-stream" \
  --data-binary @example.txt

# Get the list of files for this record
$ curl -X GET http://localhost:5000/api/records/1/files/

# Download the file named ``example.txt`` of this record
```

(continues on next page)

(continued from previous page)

```
$ curl -X GET http://localhost:5000/api/records/1/files/example.txt \
-o example.txt
```

Invenio-Records-Files provides the same REST endpoints for bucket and objects available in [Invenio-Files-REST](#), by implicitly injecting the record's bucket ID to the request.

For example given the following configuration:

```
# Invenio-Records-REST
RECORDS_REST_ENDPOINTS = {
    recid: {
        # ...,
        item_route='/records/<pid(recid):pid_value>',
        #...,
    },
    docid: {
        # ...,
        item_route='/documents/<pid(docid):pid_value>',
        #...,
    }
}

# Invenio-Records-Files
RECORDS_FILES_REST_ENDPOINTS = {
    'RECORDS_REST_ENDPOINTS': {
        'recid': '/files',
        'docid': '/doc-files',
    },
    'DEPOSIT_REST_ENDPOINTS': {
        'depid': '/deposit-files',
    }
}
```

You can access the files of a record with PID 1 using the URL `/api/records/1/files` or of a document with PID 123 using the URL `/api/documents/123/doc-files`.

You can access a specific file, for instance `example.txt`, with the following URL `/api/records/1/files/example.txt`.

Invenio-Records-Files endpoint offers the same functionality provided by [Invenio-Files-REST API](#). More information about handling files through the REST API can be found [here](#).

1.3.9 Integration with Invenio-Records-UI

If you are using [Invenio-Records-UI](#), you can easily add new views by defining new endpoints into your `RECORDS_UI_ENDPOINTS` configuration. In particular, you can add the `file_download_ui` endpoint:

```
RECORDS_UI_ENDPOINTS = dict(
    recid=dict(
        # ...
        route='/records/<pid_value>/files/<filename>',
        view_imp='invenio_records_files.utils:file_download_ui',
        record_class='invenio_records_files.api:Record',
    )
)
```


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

2.1.1 Record API

2.1.2 Utilities

2.1.3 Models

2.1.4 Links

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-records-files/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-Records-Files could always use more documentation, whether as part of the official Invenio-Records-Files 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-records-files/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* for local development.

1. Fork the *invenio* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/invenio-records-files.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-records-files
$ cd invenio-records-files/
$ 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 "Your detailed description of your changes."
$ 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.com/inveniosoftware/invenio-records-files/pull_requests and make sure that the tests pass for all supported Python versions.

3.2 Changes

Version 1.2.1 (released 2019-11-21)

- increase invenio-files-rest version to provide signals for deletion and uploading files

Version 1.2.0 (released 2019-11-19)

- Adds link factory for files and record
- Fixes the blueprints building

Version 1.1.1 (released 2019-07-31)

- Fixes missing entry point definition for the extension, causing the extension and config not to be loaded.
- Fix issue with when used with Flask-Talisman.

Version 1.1.0 (released 2019-07-29)

- Backward incompatible changes to API.

Version 1.0.0 (released 2019-07-23)

- Initial public release.

3.3 License

MIT License

Copyright (C) 2016-2019 CERN.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Note: In applying this license, CERN does not waive the privileges and immunities granted to it by virtue of its status as an Intergovernmental Organization or submit itself to any jurisdiction.

3.4 Contributors

- Alexander Ioannidis
- Alizee Pace
- Dinos Kousidis
- Jacopo Notarstefano
- Javier Delgado
- Jiri Kuncar
- Krzysztof Nowak
- Lars Holm Nielsen
- Leonardo Rossi
- Nikos Filippakis
- Sami Hiltunen
- Tibor Simko

i

`invenio_records_files`, [4](#)

`invenio_records_files.config`, [3](#)

I

`invenio_records_files` (*module*), 4
`invenio_records_files.config` (*module*), 3

R

`RECORDS_FILES_REST_ENDPOINTS` (*in module invenio_records_files.config*), 3