# Invenio-SWORD

***Release 1.0.0a10***

**CottageLabs <hello@cottagelabs.com>**

**Jul 15, 2020**

# CONTENTS:

CHAPTER

ONE

# INSTALLATION

## 1.1 With InvenioRDM

The simplest way to get set up with invenio-sword is through InvenioRDM. If you don't already have an InvenioRDM instance, follow the InvenioRDM installation instructions

First, add "invenio-sword" to your `Pipfile` with:

```
$ pipenv install invenio-sword
```

Then you can run your InvenioRDM instance as normal with *invenio-cli containerize*.

## 1.2 Outside InvenioRDM

If you are using invenio-app, then invenio-sword will be discovered automatically through its entrypoints. Otherwise, add `InvenioSword` to your API application:

```
from invenio_sword import InvenioSword

api_app = Flask("api-app")
# ...
InvenioSword(api_app)
```

Because invenio-sword extends invenio-deposit, you will also need to ensure that the invenio-deposit search mappings are installed, either by adding an entrypoint to your project, or registering them at application creation time:

```
# Entrypoint, in setup.py
setup(
    # ...
    entry_points={
        "invenio_search.mappings": [
            # ...
            "deposits = invenio_deposit.mappings",
        ],
    },
    # ...
)

# At application creation time
from invenio_search import InvenioSearch
```

(continues on next page)

(continued from previous page)

```
app = Flask("app")
# ...
search = InvenioSearch(app)
search.register_mappings("deposits", "invenio_deposit.mappings")
```

Once all that's done, you should have a default SWORD service document endpoint at http://localhost:5000/api/sword/service-document.

CHAPTER

TWO

# CONFIGURATION

By default, invenio-sword's configuration mirrors the default configuration for invenio-deposit. It's possible to have multiple SWORD endpoints, each depositing using a different `pid_type`. The default `pid_type` is `"depid"`.

```
_PID = 'pid(depid,record_class="invenio_sword.api:SWORDDeposit")'

SWORD_ENDPOINTS = {
    name: {
        **options,
        "packaging_formats": {
            ep.name: ep.load()
            for ep in pkg_resources.iter_entry_points("invenio_sword.packaging")
        },
        "metadata_formats": {
            ep.name: ep.load()
            for ep in pkg_resources.iter_entry_points("invenio_sword.metadata")
        },
        "default_packaging_format": "http://purl.org/net/sword/3.0/package/Binary",
        "default_metadata_format": "http://purl.org/net/sword/3.0/types/Metadata",
        "record_class": "invenio_sword.api:Deposit",
        "default_media_type": "application/ld+json",
        # Permissions
        "create_permission_factory_imp": permissions.check_has_write_scope,
        "read_permission_factory_imp": permissions.check_is_record_owner,
        "update_permission_factory_imp": permissions.check_has_write_scope_and_is_
↪record_owner,
        "delete_permission_factory_imp": permissions.check_has_write_scope_and_is_
↪record_owner,
        "service_document_route": "/sword/service-document",
        "item_route": "/sword/deposit/<{}:pid_value>".format(_PID),
        "metadata_route": "/sword/deposit/<{}:pid_value>/metadata".format(_PID),
        "fileset_route": "/sword/deposit/<{}:pid_value>/fileset".format(_PID),
        "file_route": "/sword/deposit/<{}:pid_value>/file/<path:key>".format(_PID),
    }
    for name, options in DEPOSIT_REST_ENDPOINTS.items()
}
```

## 2.1 Permissions configuration

You currently need to configure invenio-files-rest to allow all access to files:

```
# This allows access to files across all of invenio-files-rest
FILES_REST_PERMISSION_FACTORY = lambda *a, **kw: type(
    "Allow", (object,), {"can": lambda self: True}
)()
```

This needs a better solution in invenio-sword.

CHAPTER

THREE

# FEATURES

invenio-sword supports the following:

- Deposit of metadata, individual files and packages
- Support for In-Progress deposits
- By-reference deposit, with and without dereferencing

See the reference implementation status page for further details on implementation progress.

CHAPTER

FOUR

# DEVELOPING INVENIO-SWORD

## 4.1 Pre-commit checks

invenio-sword uses pre-commit for automated checking and reformatting on every commit. This includes:

- using black for consistent code style
- using mypy for static type checking

These checks are also run in CI.

You should ensure you have pre-commit installed, by e.g.

```
pip install --user pre-commit
```

Once you have cloned the invenio-sword repository, you should install the pre-commit hook:

```
git clone https://github.com/swordapp/invenio-sword.git
cd invenio-sword
pre-commit install
```

CHAPTER

FIVE

# API REFERENCE

## 5.1 Views

### 5.1.1 Blueprint creation

### 5.1.2 View classes

invenio-sword is an implementation of the SWORD deposit protocol for the Invenio ecosystem, building on the invenio-deposit component.

CHAPTER

SIX

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## i

# INDEX

## I