# PresQT

Center For Research Computing

Oct 21, 2021

# CONTENTS

`PresQT (Preservation Quality Tool)` is an open-source tool with RESTful services to improve Preservation and Re-use of Research Data and Software.

---

**Note:** Development is underway by the development team in the Center for Research Computing at Notre Dame. This documentation will grow/be changed throughout the life of the PresQT project. Readers should expect that pages will often be incomplete and/or move as features are actively being developed and implemented. Please send any **feedback** to the content presented here to Noel Recla **nrecla@nd.edu**

---

More information can be found here https://presqt.crc.nd.edu/

**Current Target Integrations**:

| **Target** | **Collection** | **Search** | **Detail** | **Download** | **Upload** | **Transfer In [Targets]** | **Transfer Out [Targets]** | **Hash Algorithms** | **Keyword Get** | **Keywords Upload** |
|---|---|---|---|---|---|---|---|---|---|---|
| OSF | | | | | | [Github, CurateND, Zenodo, GitLab, FigShare] | [Github, Zenodo, GitLab, FigShare] | [sha256, md5] | | |
| curateND | | | | | | | [OSF, Github, Zenodo, GitLab, FigShare] | [md5] | | |
| Github | | | | | | [OSF, CurateND, Zenodo, GitLab, FigShare] | [OSF, Zenodo, GitLab, FigShare] | [ ] | | |
| Zenodo | | | | | | [OSF, Github, CurateND, GitLab, FigShare] | [OSF, Github, GitLab, FigShare] | [md5] | | |
| GitLab | | | | | | [OSF, Github, CurateND, Zenodo, FigShare] | [OSF, GitHub, Zenodo, FigShare] | [sha256] | | |
| FigShare | | | | | | [OSF, Github, CurateND, Zenodo, Gitlab] | [OSF, Github, Gitlab, Zenodo] | [md5] | | |

**Current Service Integrations**:

| **Service** | **Functionality** |
|---|---|
| EaaSI | Send resources from a PresQT server to EaaSI to generate an emulation proposal |
| Keyword Enhancement | Suggest/add keywords to existing keywords |
| FAIRshare Evaluator | Evaluate the FAIR-ness of a resource |
| FAIRshake Assessment | Manually assess the FAIRness of a resource |

CHAPTER

ONE

# ARCHITECTURE/INFRASTRUCTURE

## 1.1 Development Environments

PresQT uses Docker throughout its pipeline to make it as easy as possible for newcomers to the project to get up and running with PresQT services. This was accomplished by creating an easy to use development environment using Docker compose.

The diagram below illustrates how we are using Docker Compose to create a constellation of 2 containers on developer machines representing the two essential components of PresQT:

- Nginx
  - Serves as a security layer for incoming requests to PresQT
  - Can also serve as a load balancer in the future
- Django/Gunicorn
  - After passing through the Nginx layer, this container processes API requests from users and then takes the necessary actions to fulfill the user's requests by communicating with partner services via the their own APIs.

INSERT IMAGE HERE

## 1.2 QA/Production Deployments

Unsurprisingly, there is a significant overlap between the developer setup and the QA/Production deployment architecture. The following is how they will vary:

- One machine (the "Web Server") will be the host for the Nginx container and one or more identical Django containers that will respond to APIs requests from client researchers in a load balanced manner.

INSERT IMAGE HERE

# CHAPTER TWO

# DEVELOPMENT ENVIRONMENT SETUP

## 2.1 Prerequisites

- Local installation of Docker for Mac/Windows/Linux
- Knowledge of Git procedures
- Knowledge of setting environment variables
- Knowledge of `docker-compose` utility.

## 2.2 Local Development Environment Setup

1. Clone the repo to your local machine in the desired folder location:

```
$ git clone https://github.com/ndlib/presqt.git
```

2. Export **required** ENV_VARS:
   - **ENVIRONMENT**: Should be either `production` or `development`
   - **SECRET_KEY**: A Django "secret key" value.

```
# Example Exportation
$ export ENVIRONMENT=development
$ export SECRET_KEY=y4xgryt7ex9g+4mcs4=^sg5afp31z#=94eb6=6o6l61o=a31y_h
```

3. Export **optional** ENV_VARS for testing:
   - **CURATE_ND_TEST_TOKEN**: The test token for Curate's API.
   - **GITHUB_TEST_USER_TOKEN**: The test token for GitHub's API.
   - **OSF_TEST_USER_TOKEN**: The test token for OSF's API.
   - **OSF_PRIVATE_USER_TOKEN**: The private test token for OSF's API.
   - **OSF_UPLOAD_TEST_USER_TOKEN**: The upload test token for OSF's API.
   - **OSF_PRESQT_FORK_TOKEN**: The PresQT fork user test token for OSF's API.
   - **ZENODO_TEST_USER_TOKEN**: The test token for Zenodo's API.
   - **GITLAB_TEST_USER_TOKEN**: The test token for GitLab's API.
   - **FIGSHARE_TEST_USER_TOKEN**: The test token for FigShare's API.

**Note:** Contact an administrator to get the target test tokens.

4. Execute `docker-compose` up within the repo's base folder.

```
$ docker-compose up --build
```

5. Navigate to https://localhost/api_v1/ in your browser.

## 2.3 Cron Container

There is now a third docker container that is responsible for running clean up tasks at specified times. It has been implemented in development to run the *delete_outdated_mediafiles* command every 15 minutes. The command has also been altered slightly to delete any mediafiles held in these directories when you are in a development environment. The command is set to run daily at 4:30am for our other servers.

CHAPTER

THREE

# AUTHENTICATION/AUTHORIZATION

PresQT will not have the ability to create a 'session' for the user based on authentication. It will be expecting tokens to be passed through the header of the request. When retrieving items it expects 'presqt-source-token' to be in the header. When depositing an item it expects 'presqt-destination-token' to be in the header.

## 3.1 Target Token Instructions

### 3.1.1 Open Science Framework

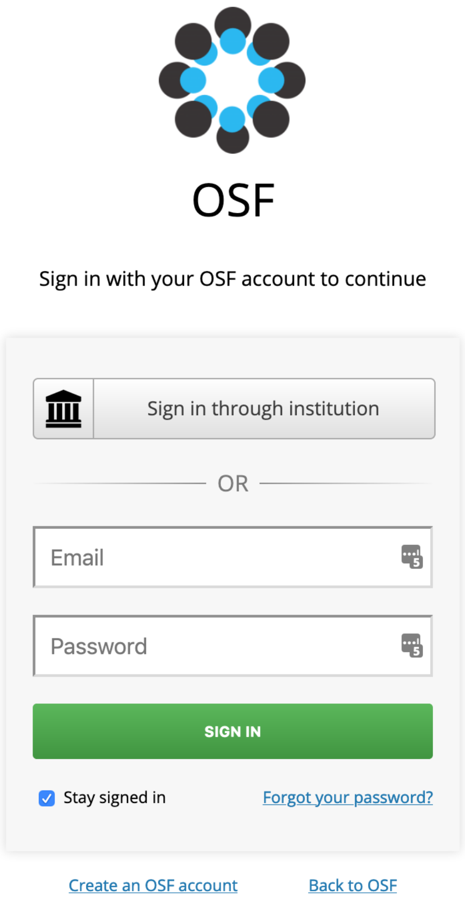
1. Navigate to and login to your account.
2. Upon logging in, click on your username in the top right corner and then click on `Settings`.
3. Once in`` `Settings`, click on `Personal Access Tokens` in the left hand menu.
4. Click on `Create token`.
5. Create a token name and select all scope options. Then press `Create token`.
6. Make sure you copy this token somewhere securely, this will be the only time it is shown to you.

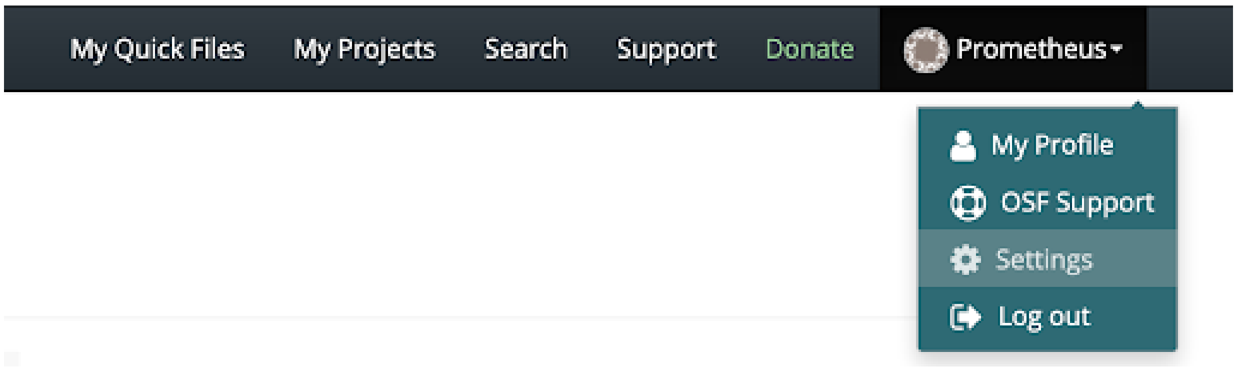
### 3.1.2 CurateND

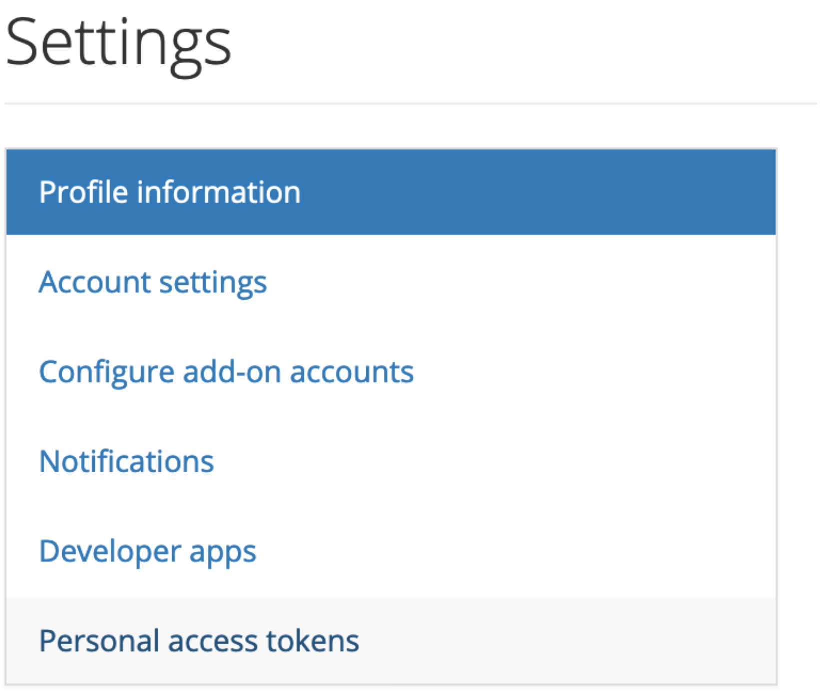
1. Navigate to and login to your account.
2. In the top right corner, select `Manage` and then click on `API Access Tokens`.
3. Click on `Create New Token`.
4. Make sure you copy this token somewhere securely.

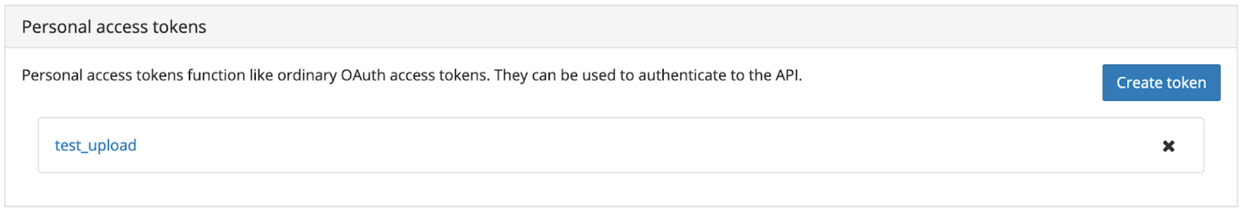
### 3.1.3 GitHub

1. Navigate to and login to your account.
2. In the top right corner, select your profile picture and then click on `Settings`.
3. In the bottom left of your settings, select `Developer Settings`.
4. On the left hand side of this screen, select `Personal Access Tokens`.
5. Click on `Generate New Token`.
6. Add a note about what the token will be used for, and select all scopes. Then select `Generate Token`.

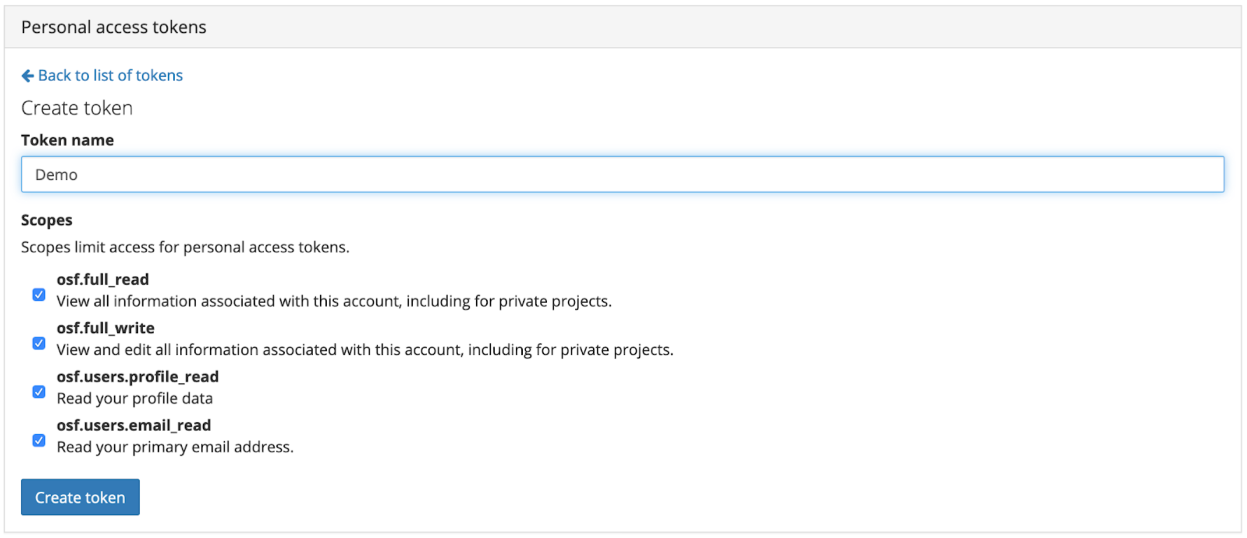

OSF
Sign in with your OSF account to continue
Sign in through institution
OR
Email
Password
SIGN IN
Stay signed in
Forgot your password?
Create an OSF account
Back to OSF


My Quick Files
My Projects
Search
Support
Donate
Prometheus
My Profile
OSF Support
Settings
Log out





Personal access tokens function like ordinary OAuth access tokens. They can be used to authenticate to the API.



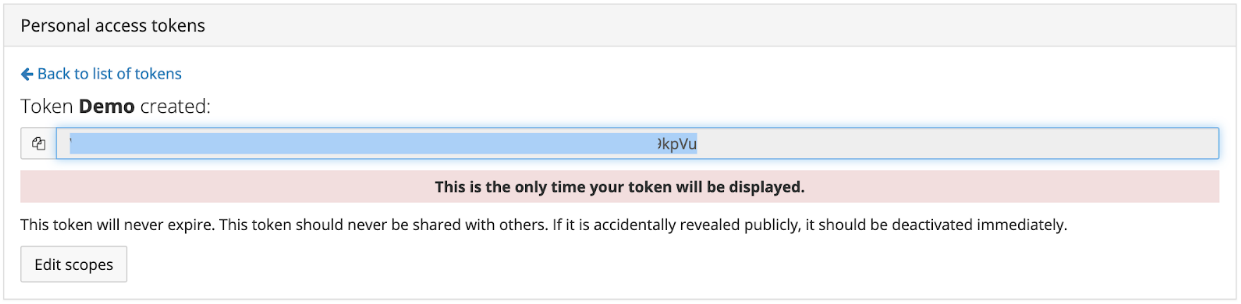
**Token name**

Demo

**Scopes**

Scopes limit access for personal access tokens.

- [✓] **osf.full_read**
  View all information associated with this account, including for private projects.
- [✓] **osf.full_write**
  View and edit all information associated with this account, including for private projects.
- [✓] **osf.users.profile_read**
  Read your profile data
- [✓] **osf.users.email_read**
  Read your primary email address.



Personal access tokens

Back to list of tokens

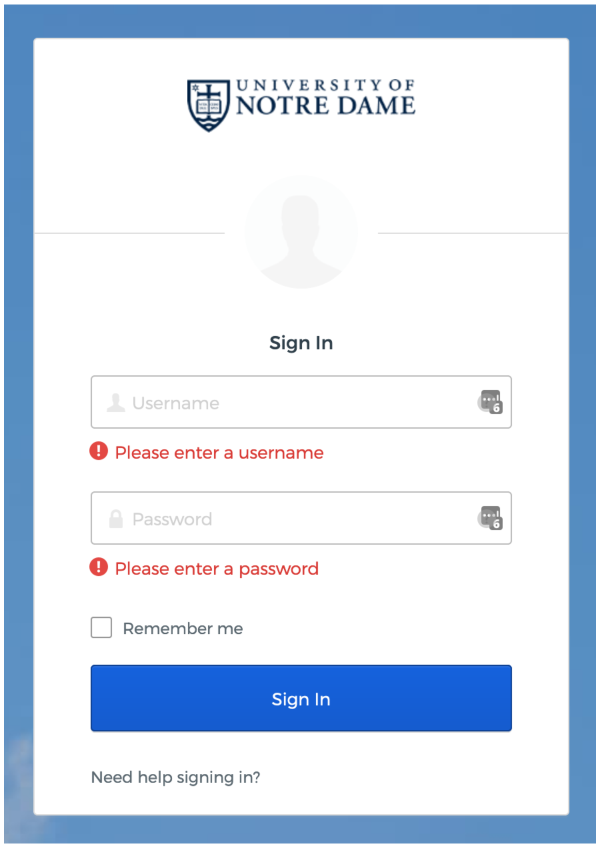
Token **Demo** created:

…kpVu

**This is the only time your token will be displayed.**

This token will never expire. This token should never be shared with others. If it is accidentally revealed publicly, it should be deactivated immediately.

Edit scopes



UNIVERSITY OF NOTRE DAME

Sign In

Username

Please enter a username

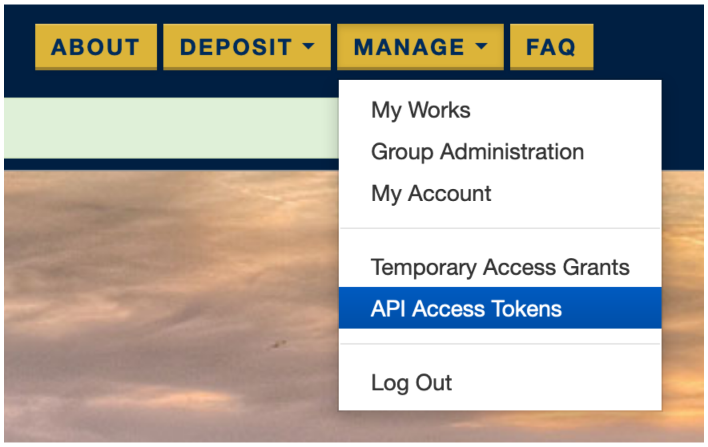
Password

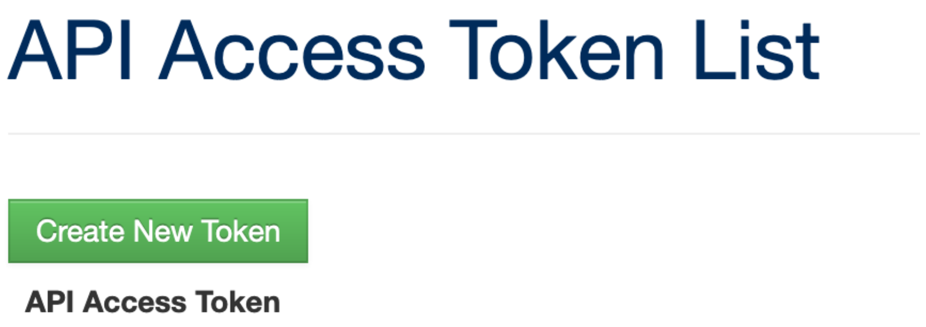
Please enter a password

Remember me

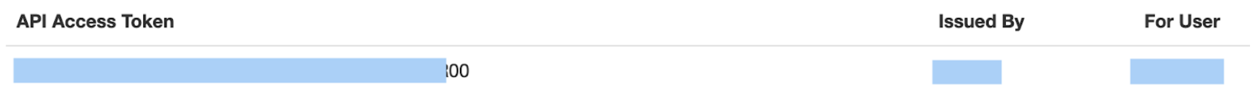
Sign In

Need help signing in?

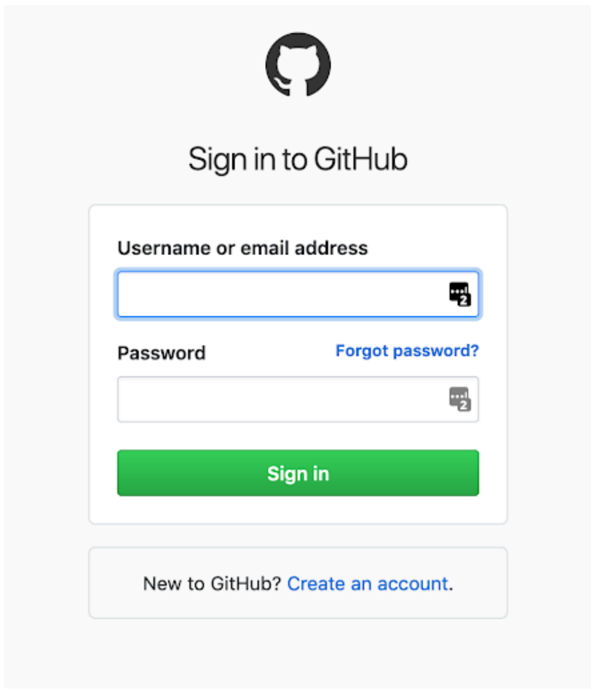


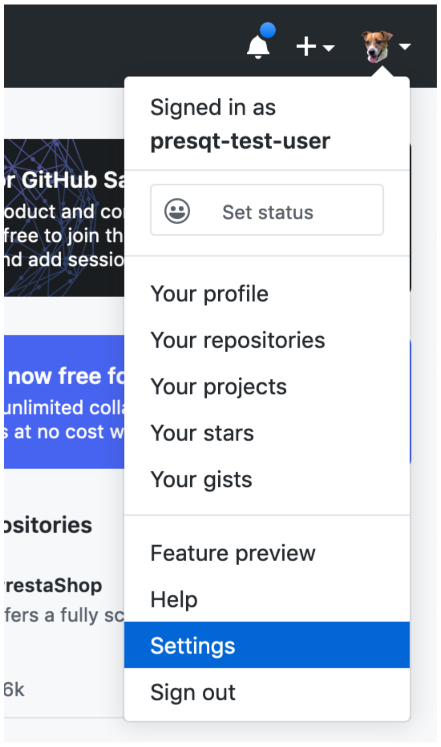

API Access Token List

API Access Token



| API Access Token | Issued By | For User |
|---|---|---|
| [redacted]00 | [redacted] | [redacted] |


Sign in to GitHub



7. Make sure you copy this token somewhere securely, this will be the only time it is shown to you.

### 3.1.4 Zenodo

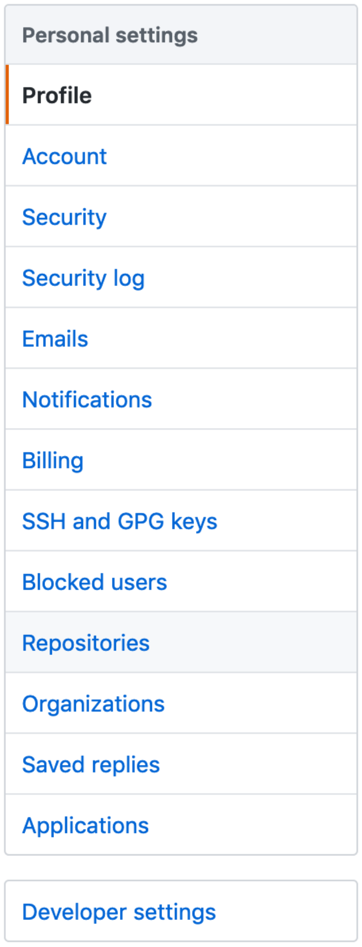
1. Navigate to and login to your account.
2. In the top right corner, select your username and then click on `Applications`.
3. In the `Personal access tokens` section, click on `New token`.
4. Give the token a name and select all scopes, then click `Create`.
5. Make sure you copy this token somewhere securely, this will be the only time it is shown to you.

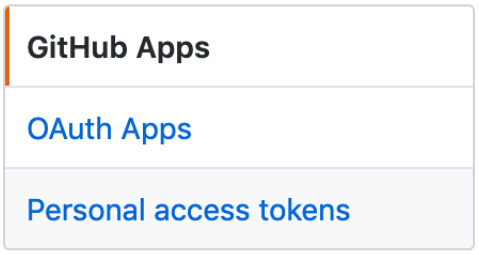
### 3.1.5 GitLab

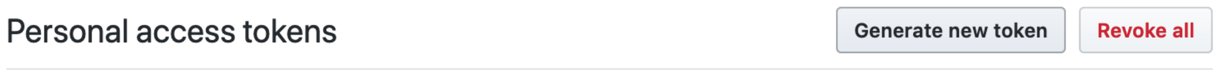
1. Navigate to and login to your account.
2. In the top right corner, select your username and then click on `Settings`.
3. In the left hand menu, select `Access Tokens`.
4. Give the token a name and select all scopes, then click `Create personal access token`.
5. Make sure you copy this token somewhere securely, this will be the only time it is shown to you.

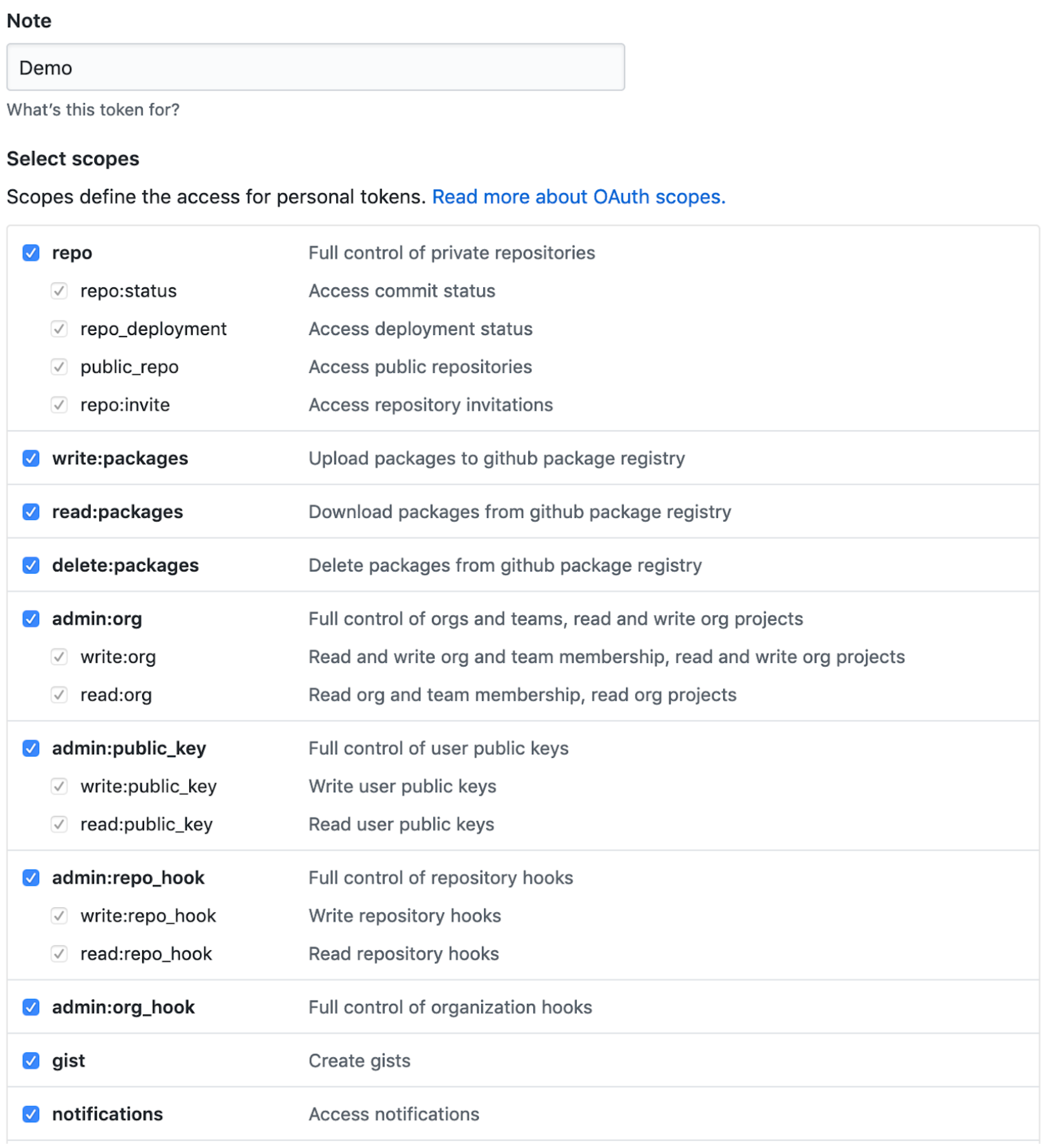
### 3.1.6 FigShare

1. Navigate to and login to your account.
2. In the top right corner, select your username and then click on `Applications`.


Personal settings
Profile
Account
Security
Security log
Emails
Notifications
Billing
SSH and GPG keys
Blocked users
Repositories
Organizations
Saved replies
Applications
Developer settings


GitHub Apps
OAuth Apps
Personal access tokens


Personal access tokens
Generate new token
Revoke all



**Note**

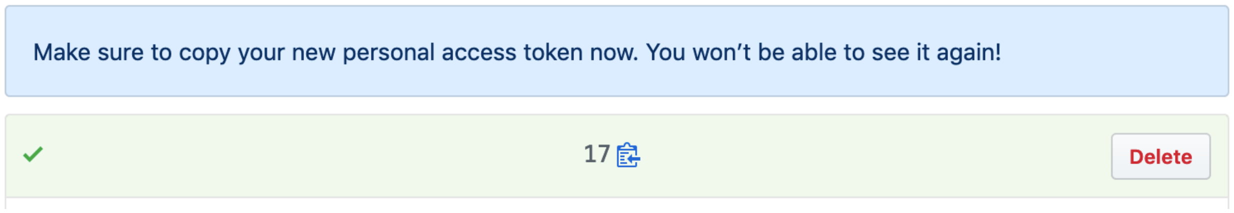
Demo

What's this token for?

**Select scopes**

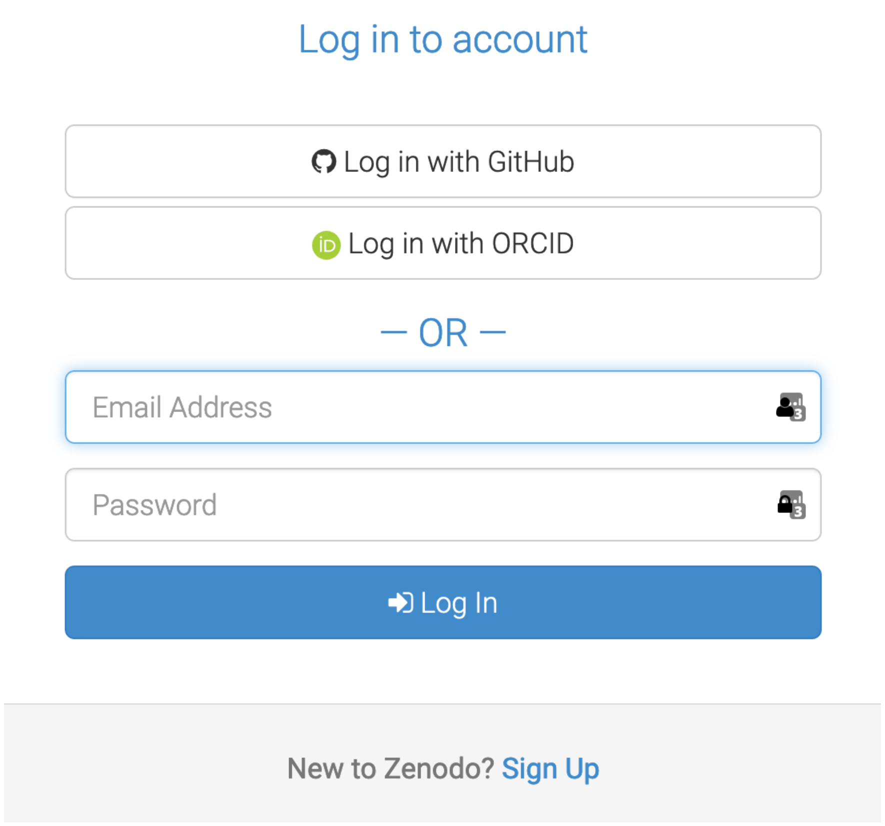
Scopes define the access for personal tokens. Read more about OAuth scopes.

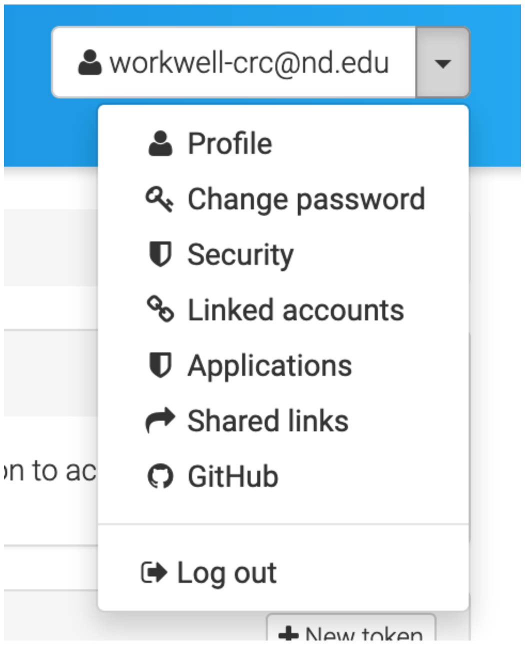
| | |
|---|---|
| ✓ **repo** | Full control of private repositories |
| ✓ repo:status | Access commit status |
| ✓ repo_deployment | Access deployment status |
| ✓ public_repo | Access public repositories |
| ✓ repo:invite | Access repository invitations |
| ✓ **write:packages** | Upload packages to github package registry |
| ✓ **read:packages** | Download packages from github package registry |
| ✓ **delete:packages** | Delete packages from github package registry |
| ✓ **admin:org** | Full control of orgs and teams, read and write org projects |
| ✓ write:org | Read and write org and team membership, read and write org projects |
| ✓ read:org | Read org and team membership, read org projects |
| ✓ **admin:public_key** | Full control of user public keys |
| ✓ write:public_key | Write user public keys |
| ✓ read:public_key | Read user public keys |
| ✓ **admin:repo_hook** | Full control of repository hooks |
| ✓ write:repo_hook | Write repository hooks |
| ✓ read:repo_hook | Read repository hooks |
| ✓ **admin:org_hook** | Full control of organization hooks |
| ✓ **gist** | Create gists |
| ✓ **notifications** | Access notifications |

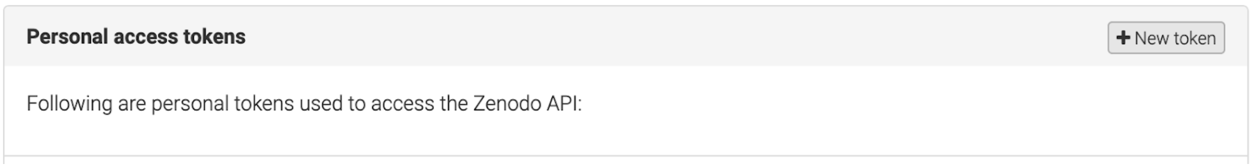


Make sure to copy your new personal access token now. You won't be able to see it again!

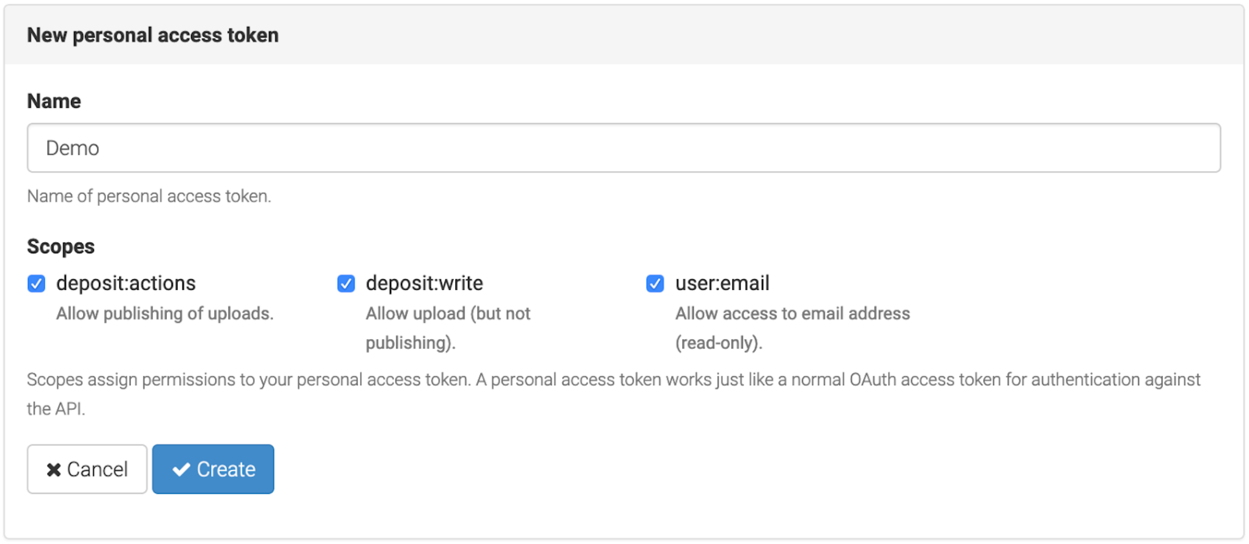
✓ 17







Following are personal tokens used to access the Zenodo API:



**New personal access token**

**Name**

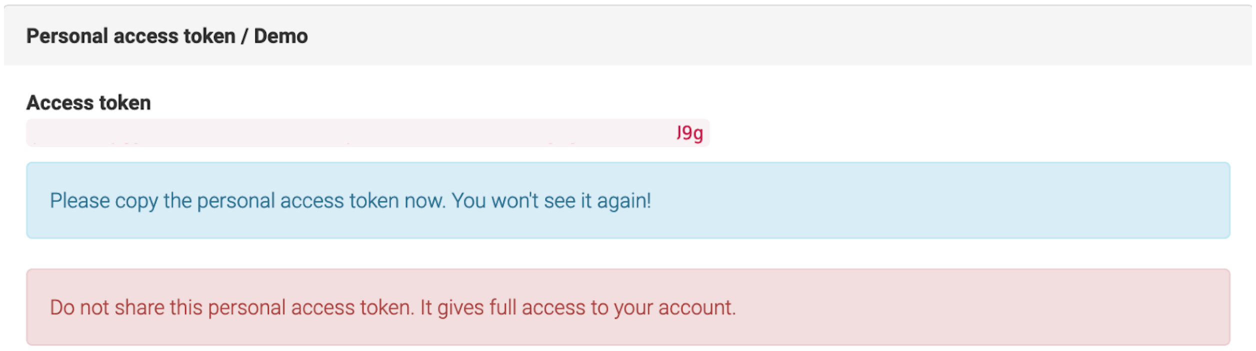
Demo

Name of personal access token.

**Scopes**

- [✓] deposit:actions — Allow publishing of uploads.
- [✓] deposit:write — Allow upload (but not publishing).
- [✓] user:email — Allow access to email address (read-only).

Scopes assign permissions to your personal access token. A personal access token works just like a normal OAuth access token for authentication against the API.



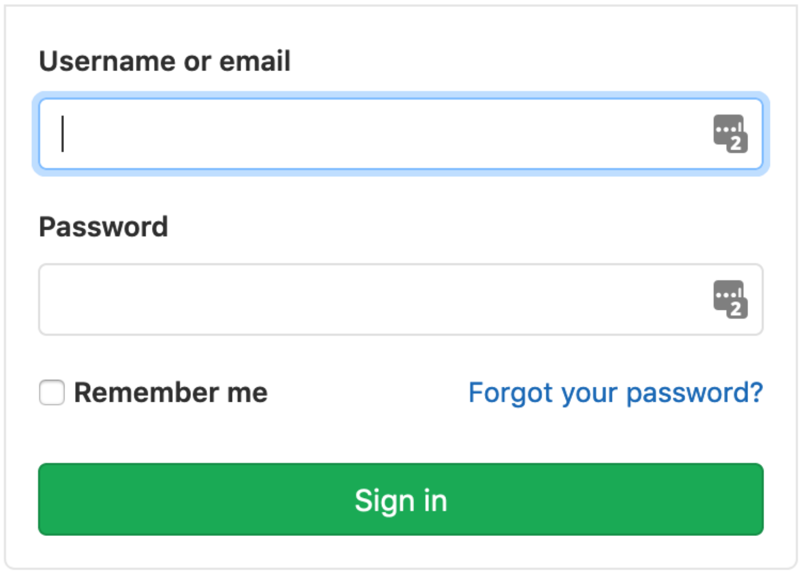
**Personal access token / Demo**

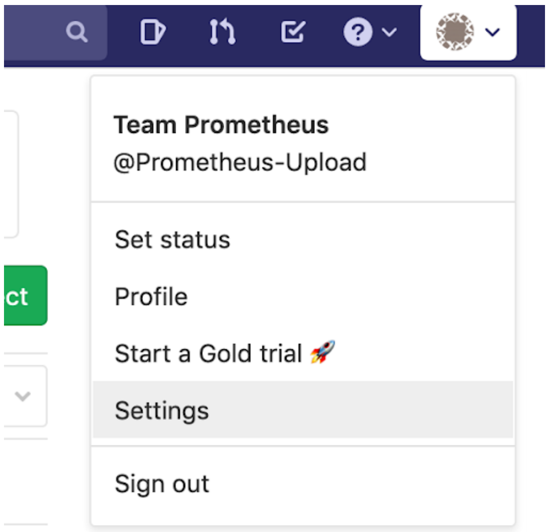
**Access token**

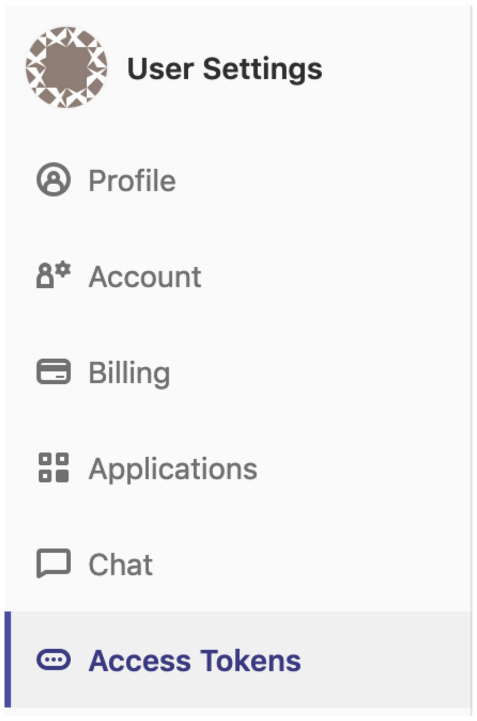
J9g

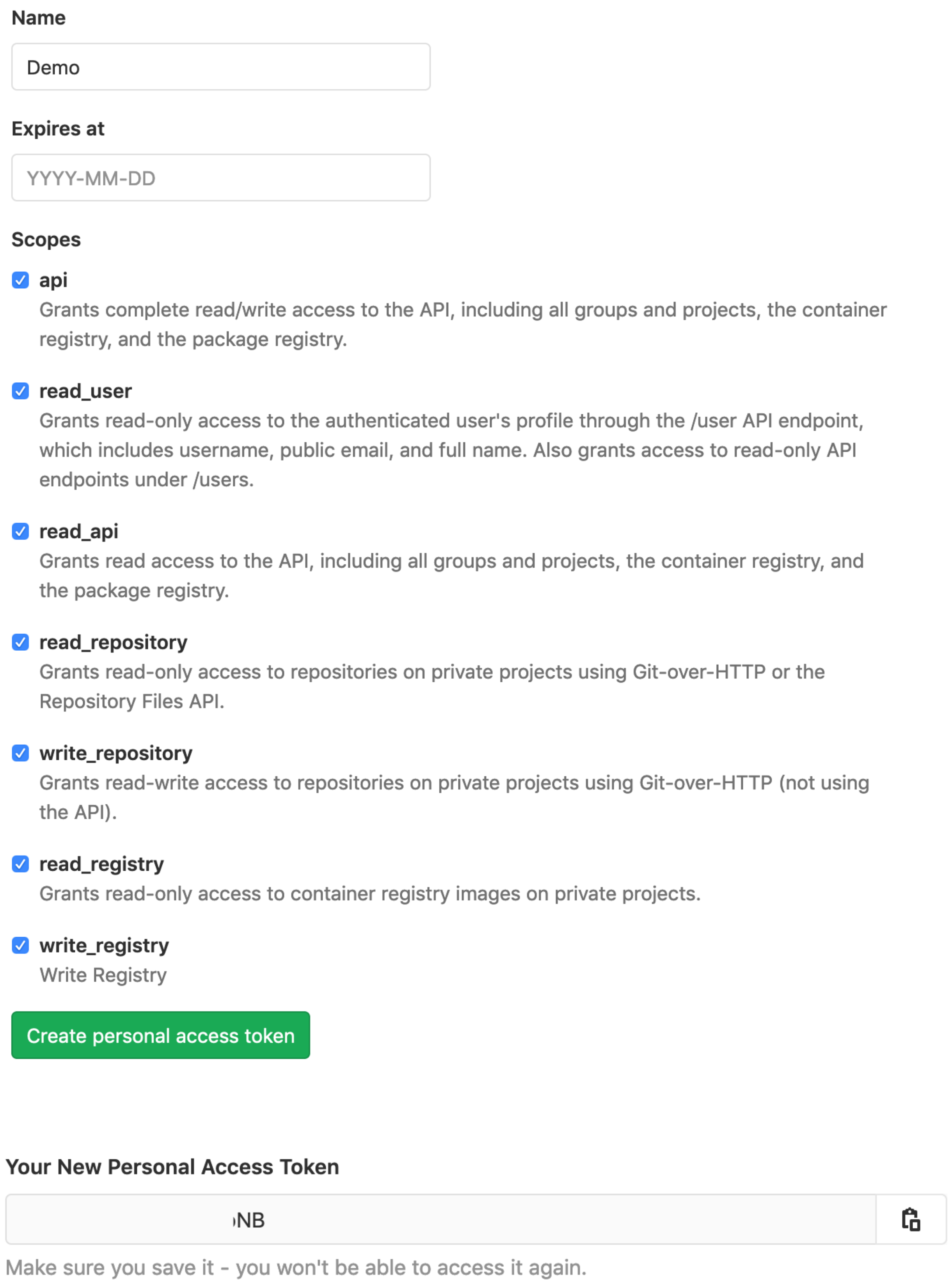
Please copy the personal access token now. You won't see it again!

Do not share this personal access token. It gives full access to your account.




Team Prometheus
@Prometheus-Upload
Set status
Profile
Start a Gold trial 🚀
Settings
Sign out
ct


User Settings
Profile
Account
Billing
Applications
Chat
Access Tokens



**Name**

Demo

**Expires at**

YYYY-MM-DD

**Scopes**

[✓] **api**
Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.

[✓] **read_user**
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.

[✓] **read_api**
Grants read access to the API, including all groups and projects, the container registry, and the package registry.

[✓] **read_repository**
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.

[✓] **write_repository**
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).

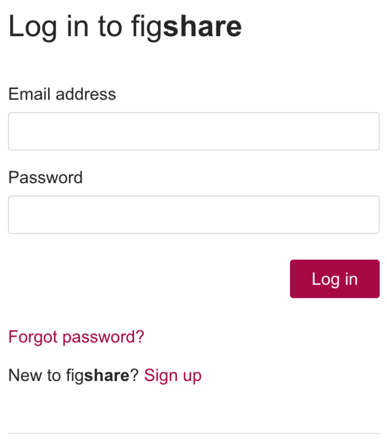
[✓] **read_registry**
Grants read-only access to container registry images on private projects.

[✓] **write_registry**
Write Registry

**Your New Personal Access Token**

NB

Make sure you save it - you won't be able to access it again.


Log in to figshare

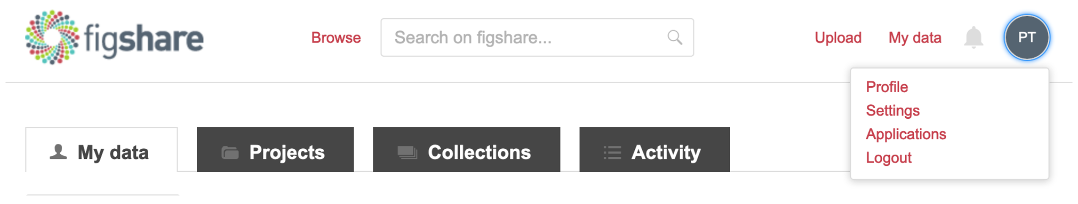
Email address

Password

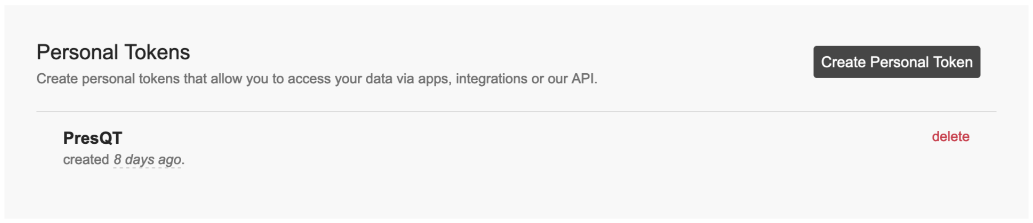
Log in

Forgot password?

New to figshare? Sign up



3. Scroll down to the bottom of the screen, and click `Create Personal Token`.


Personal Tokens

Create personal tokens that allow you to access your data via apps, integrations or our API.

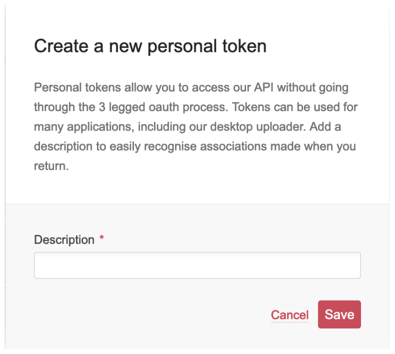
Create Personal Token

**PresQT**

created *8 days ago*.

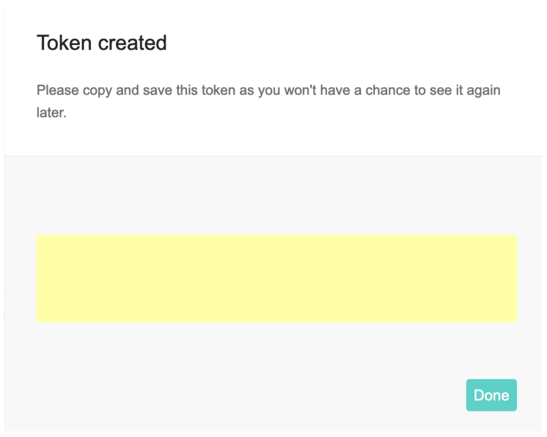
delete

4. Give the token a description (name), then click `Save`.
5. Make sure you copy this token somewhere securely, this will be the only time it is shown to you.


Create a new personal token

Personal tokens allow you to access our API without going through the 3 legged oauth process. Tokens can be used for many applications, including our desktop uploader. Add a description to easily recognise associations made when you return.

Description *


Token created

Please copy and save this token as you won't have a chance to see it again later.

CHAPTER

# FOUR

# USER NOTES

## 4.1 Transfer Details

| Target | As Source | As Destination |
|---|---|---|
| OSF | Only provides checksums for OSF Storage files<br>Keywords written to 'Tags' attribute | Writes PRESQT_FTS_METADATA.json file<br>Keywords written to 'Tags' attribute.<br>Stores all resources in OSF Storage |
| curateND | Provides checksums for all files | N/A |
| Github | Does not provide checksums for files<br>Keywords written to 'Topics' attribute | Writes PRESQT_FTS_METADATA.json file<br>Does not provide checksums for files<br>Keywords written to 'Topics' attribute |
| Zenodo | Provides checksums for all files<br>Keywords written to 'Keywords' attribute | Writes PRESQT_FTS_METADATA.json file<br>Resources will be written in BagIt format as a ZIP file<br>Keywords written to 'Keywords' attribute |
| GitLab | Provides checksums for all files<br>Keywords written to 'Tag List' attribute | Writes PRESQT_FTS_METADATA.json file<br>Keywords written to 'Tag List' attribute |
| Figshare | Provides checksums for all files<br>Keywords written to 'Tags' attribute | Writes PRESQT_FTS_METADATA.json file<br>Resources will be written in BagIt format as a ZIP file<br>Keywords written to 'Tags' attribute |

CHAPTER

# FIVE

# DEVELOPER NOTES

## 5.1 Testing

A high code coverage percentage has been maintained with unit and integration tests for all code using a package called Coverage (https://coverage.readthedocs.io/en/v4.5.x/) to track code coverage.

To run unit tests without using Coverage:

```
$ python manage.py test
```

To run unit tests using Coverage with comprehensive code coverage report generated into an HTML file:

```
coverage run manage.py test && coverage combine && coverage html
```

**Note:** This command will generate a directory that is ignored by Git via our .gitignore file. To see the code coverage open the file /coverage_html/index.html in a browser.

**Note:** Coverage options are specified in a configuration file called .coveragerc. This is where you would add files/directories you want to omit from the Coverage report.

**Note:** 'coverage combine' will take the coverage files created for multiprocesses (located in the base directory) and will combine them with the main coverage files . If a test using multiprocessing fails these coverage files will remain and must be deleted manually.

We also tried to split unit and integration tests up between core PresQT code and Target code. Tests that cover core code reside in `presqt/api_v1/tests/` while target tests that cover target functions reside in `presqt/targets/{target_name}/tests/`.

**Attention:** All tests require their corresponding target tokens to be stored as environment variables since these tokens can not be stored publicly. Contact an administrator for access to these.

## 5.2 Docker Commands

To rebuild the docker container after a new package has been added to the requirements files:

```
$ docker-compose up --build
```

Run the following command for an *interactive -i terminal -t* for this container:

```
$ docker exec -i -t presqt_presqt_django_1 /bin/ash
```

## 5.3 Updating Documentation

As the project grows we encourage developers to add documentation. PresQT documentation is built using Sphinx and ReadtheDocs.

When documentation is added you should just need to run while in the `/docs` directory:

```
$ make clean
$ make html
```

Otherwise reference Sphinx documentation for more information on adding documentation, https://www.sphinx-doc.org/en/master/usage/quickstart.html.

## 5.4 GitHub Differences

There is a slight difference in how we have implemented GitHub as opposed to other partners. Due to the way GitHub's API handles files, there is no way for us to hit an endpoint with a given id. The only way to navigate to a files endpoint is to know the associated GitHub username and repository title. We have decided to make our own unique id's for these items by combining the repo_id and the path to the file/dir.

Example of an id:

```
21387123:path2%Fto%2Ffile%2Ejpg
```

We then translate this into two get requests. The first one will be to the repo id.

Example:

```
https://api.github.com/repositories/21387123
```

From this, we can get a contents url that we can translate into a request.

Example:

```
https://api.github.com/repos/PresQT-QA-TEST/Good_Egg-PresQT2-/contents/path/to/file.
↪jpg
```

Using this custom generated id, we can hit this endpoint on PresQT to get file details.

```
https://presqt-qa.crc.nd.edu/api_v1/targets/github/resources/21387123:path%252Fto
↪%252Ffile%252Ejpg
```

# CHAPTER SIX

# TARGET INTEGRATION

The goal of PresQT is to make it as simple as possible for a new target to integrate itself with the PresQT services. Below are lists of code actions to take when integrating a target.

## 6.1 Target Endpoints

'Targets' are providers the PresQT API will connect to such as OSF, CurateND, HubZero, etc. Since PresQT doesn't have a database, the Targets' information will be held in a JSON file located in `/presqt/specs/targets.json`. You must add data to this file to integrate with PresQT.

### 6.1.1 Target Collection/Details

1. Add your target dictionary to the file `presqt/specs/targets.json`

   **Target JSON Details:**

| Key | Type | Description |
|---|---|---|
| name | str | Name of the Target. This will be used as path parameters in the URL |
| readable_name | str | Human readable name of the Target for the front end |
| status_url | str | Url which is 200 OK if the API works. |
| token_url | str | Url where users can create their API tokens. |
| supported_actions | array | Actions the target supports. Only make actions true when action is working |
| resource_collection | bool | Get all top level resources for the user in this target |
| resource_detail | bool | Get an individual resource's details |
| resource_download | bool | Download a resource |
| resource_upload | bool | Upload a resource |
| resource_transfer_in | bool | Transfer a resource in to the target |
| resource_transfer_out | bool | Transfer a resource out of the target |
| supported_transfer_partners | dict | Targets this target can transfer in and out of |
| transfer_in | array | Targets this target can accept transfers from |
| transfer_out | array | Targets this target can transfer to |
| supported_hash_algorithms | array | The hash algorithms supported by the target |
| infinite_depth | bool | Does the target support an infinite depth hierarchy? |
| search_parameters | array | Which search parameters does the target support? options: [general, title, id, author] |
| keywords | bool | Fetch keywords |
| keywords_upload | bool | Upload keywords to the target specific keyword attribute. |

**Target JSON Example:**

```
{
    "name": "osf",
    "readable_name": "OSF",
    "status_url": "https://api.osf.io/v2/nodes/",
    "token_url": "https://osf.io/settings/tokens",
    "supported_actions": {
        "resource_collection": true,
        "resource_detail": true,
        "resource_download": true,
        "resource_upload": true,
        "resource_transfer_in": true,
        "resource_transfer_out": true,
        "keywords": true,
        "keywords_upload": true
    },
    "supported_transfer_partners": {
        "transfer_in": ["github", "curate_nd"],
        "transfer_out": ["github"]
    },
```

(continues on next page)

(continued from previous page)

```
    "supported_hash_algorithms": ["sha256", "md5"],
    "infinite_depth": true,
    "search_parameters": ["title", "id", "general", "author"]
}
```

There is a management command that will validate `targets.json` that can be run after you add your target. It can be run manually with:

```
$ python manage.py validate_target_json
```

Otherwise the same management command is run when `docker-compose up` runs. If the validation fails then it does not allow the docker containers to be spun up.

2. Add your target directory inside `presqt/targets/`
   - Your target integration functionality will exist here.

## 6.2 Resource Endpoints

### 6.2.1 Resource Collection

Targets that integrate with the Resources Collection API Endpoint must have a function that returns a specifically structured dataset.

1. Update your target in `presqt/specs/targets.json` by setting `supported_actions.resource_collection` to `true`.

2. Add a function to return the resource collection inside of your target directory.
   - If you would like to keep your file/function names consistent with what already exists add this function at `presqt/targets/<target_name>/functions/fetch/<target_name>_fetch_resources()`
   - The function must have the following parameters **in this order**:

| | | |
|---|---|---|
| token | str | User's token for the target |
| query_parameter | str | The query_parameter parameters passed to the API View |

   - The function must return the following **in this order**:

| | | |
|---|---|---|
| resources | list | list of Python dictionaries for each top level resource |
| pages | dict | dictionary of pagination details |

**Resource dictionary details:**

| | | |
|---|---|---|
| kind | str | Type of Resource<br>Options: [container, item] |
| kind_name | str | Target specific name for that kind<br>For example OSF kind_names are: [project, folder, file] |
| container | str | ID of the container for the resource.<br>For example if the resource is a file in a folder then the **container** value would be the ID of the folder<br>Can be None if the resource has no container |
| id | str | ID of the resource |
| title | str | Title of the resource |

**Page dictionary details:**

| | | |
|---|---|---|
| first_page | str | The first page number |
| previous_page | str | The previous page number |
| next_page | str | The next page number |
| last_page | str | The last page number |
| total_pages | str | The total amount of pages |
| per_page | str | The amount of resources per page |

**Example Resource Collection Function:**

```
def <your_target_name>_fetch_resources(token, query_parameter):
    # Process to obtain resource collection IF search_parameter␣
↪goes here.
    # Process to obtain resource collection goes here.
    # Variables below are defined here to show examples of␣
↪structure.
    target_resources = get_target_resources()

    resources = []
    for resource in target_resources:
           resource.append({
            'kind': 'container',
            'kind_name': 'Project',
            'id': resource.id,
            'container': None,
            'title': resource.title
        })

    # Process to obtain page numbers goes here
```

(continues on next page)

(continued from previous page)

```
    pages = {
        "first_page": '1',
        "previous_page": None,
        "next_page": None,
        "last_page": '1',
        "total_pages": '1',
        "per_page": 10
    }
    return resources, pages
```

3. Add the resource collection function to `presqt/api_v1/utilities/utils/function_router.py`
   - Follow the naming conventions laid out in this class' docstring
   - This will make the function available in core PresQT code

### 6.2.2 Resource Detail

Targets that integrate with the Resources Detail API Endpoint must have a function that returns a specifically structured dataset that represents the resource.

1. Update your target in `presqt/specs/targets.json` by setting `supported_actions.resource_detail` to `true`.
2. Add a function to return the resource details inside of your target directory.
   - If you would like to keep your file/function names consistent with what already exists add this function at `presqt/targets/<target_name>/functions/fetch/<target_name>_fetch_resource()`
   - The function must have the following parameters **in this order**:

| | | |
|---|---|---|
| token | str | User's token for the target |
| resource_id | str | ID for the resource we want to fetch |

   - The function must return the following **in this order**:

| | | |
|---|---|---|
| resource | object | Python object representing the resource requested |

**Resource dictionary details:**

| | | |
|---|---|---|
| kind | str | Type of Resource<br>Options: [container, item] |
| kind_name | str | Target specific name for that kind<br>For example OSF kind_names are: [node, folder, file] |
| id | str | ID of the resource |
| title | str | Title of the resource |
| date_created | str | Date the resource was created |
| date_modified | str | Date the resource was last modified |
| hashes | dict | Hashes of the resource in the target<br>Key must be the hash algorithm used value must be the hash itself<br>Can be an empty dict if no hashes exist |
| extra | dict | Any extra target specific data.<br>Can be an empty dict |
| children | list | A list of children resources, each child in the list must be a dictionary that follows the structure of the resource_collection dictionaries listed above. Example: [{'kind': '', 'kind_name': '', 'id': '', 'container': '', 'title': ''}] |

**Example Resource Collection Function:**

```
def <your_target_name>_fetch_resource(token, resource_id):
        # Process to obtain resource details goes here.
        # Variables below are defined here to show examples␣
↪of structure.

    resource = {
        "kind": "item",
        "kind_name": "file",
        "id": "12345",
        "title": "o_o.jpg",
        "date_created": "2019-05-13T14:54:17.129170Z",
        "date_modified": "2019-05-13T14:54:17.129170Z",
        "hashes": {
            "md5": "abca7ef057dcab7cb8d79c36243823e4",
            "sha256":
↪"ea94ce55261720c56abb508c6dcd1fd481c30c09b7f2f5ab0b79e3199b7e2b55
↪"
        },
        "extra": {
            "category": "project",
            "fork": false,
            "current_user_is_contributor": true,
            "preprint": false,
            "current_user_permissions": [
                "read",
                "write",
                "admin"
            ],
        },
        "children": []
    }
    return resource
```

3. Add the resource detail function to `presqt/api_v1/utilities/utils/function_router.py`
   - Follow the naming conventions laid out in this class' docstring

- This will make the function available in core PresQT code

## 6.3 Resource Download Endpoint

1. Update your target in `presqt/specs/targets.json` by setting `supported_actions.resource_download` to `true`.

2. Add a function to perform the resource download inside of your target directory.

   - If you would like to keep your file/function names consistent with what already exists add this function at `presqt/targets/<target_name>/functions/download/<target_name>_download_resource()`
   - The function must have the following parameters **in this order**:

| | | |
|---|---|---|
| token | str | User's token for the target |
| resource_id | str | ID for the resource we want to download |
| process_info_path | str | The path to this download's process_info_path |
| action | str | The type of action occurring |

   - The function must return a **dictionary** with the following keys:

| | | |
|---|---|---|
| resources | list | List of dictionaries containing resource data |
| empty_containers | list | List of strings identifying empty container paths.<br>They need to be specified separately because they are written separate from the file data |
| action_metadata | dict | Dictionary containing FTS metadata about the action occurring |
| extra_metadata | dict | Dictionary containing extra metadata identified by partners |

**Resource Dictionary Details**

| | | |
|---|---|---|
| file | bytes | The file contents in byte format |
| hashes | dict | Hashes of the resource in the target<br>Key must be the hash algorithm used value must be the hash itself<br>Can be an empty dict if no hashes exist |
| title | str | Title of the file |
| path | str | Path to save the file to at the destination<br>Start the path with a / |
| source_path | str | Full path of the file at the source<br>Start the path with a / |
| extra_metadata | dict | Dictionary containing any extra data to save to FTS metadata |

**Action Metadata Dictionary Details**

| | | |
|---|---|---|
| sourceUsername | str | Username of the user making the request at the source target |

**Extra Metadata Dictionary Details**

| | | |
|---|---|---|
| title | str | The title of the resource |
| creators | list | List of dictionaries containing creator info {“first_name”: ‘’, “last_name”: ‘’, “ORCID”: ‘’} |
| publication_date | str | The date the resource was published |
| description | str | A brief description of the resource |
| keywords | list | A list of associated keywords |
| license | str | The resource’s license |
| related_identifiers | list | A list of dictionaries containing identifiers {“type”: ‘doi’, “identifier”: ‘’} |
| references | str | References related to the resource |
| notes | str | Notes related to the resource |

- If you want to keep track of the progress of the download there are two functions available to do so. `update_process_info()` is for updating the total number of resources in the download and `increment_process_info()` is for updating the number of resources gathered thus far.

**Example Resource Download Function:**

```
def <your_target_name>_download_resource(token, resource_id,␣
↪process_info_path):
    # Process to download resource goes here.
    # Variables below are defined here to show examples of␣
↪structure.
    resources = [
        {
            'file': binary_file_contents,
            'hashes': {'md5': '1ab2c3d4e5f6g', 'sha256':
↪'fh3383h83fh'},
            'title': 'file.jpg',
            'path': '/path/to/file.jpg',
            'source_path': 'project_name/path/to/file.jpg',
            'extra_metadata': {
                'dateSubmitted': '2019-10-22Z',
                'creator': 'Justin Branco',
            }
        },
        {
            'file': binary_file_contents,
            'hashes': {'md5': 'zadf23fg3', 'sha256':
↪'9382hash383h'},
            'title': 'funnysong.mp3',
            'path': '/path/to/file/funnysong.mp3'
            'source_path': 'project_name/path/to/file/funnysong.
↪mp3',
            'extra_metadata': {
                'dateSubmitted': '2019-10-22Z',
                'creator': 'Justin Branco',
            }
        }
```

(continues on next page)

(continued from previous page)

```
    ]
    empty_containers = ['path/to/empty/container/', 'another/
↪empty/']
    action_metadata = {"sourceUsername": contributor_name}
    extra_metadata = {
        "title": project_info['title'],
        "creators": creators,
        "publication_date": project_info['date_created'],
        "description": project_info['description'],
        "keywords": project_info['tags'],
        "license": license,
        "related_identifiers": identifiers,
        "references": None,
        "notes": None
    }
    return {
        'resources': files,
        'empty_containers': empty_containers,
        'action_metadata': action_metadata,
        'extra_metadata': extra_metadata
    }
```

3. Add the resource download function to `presqt/api_v1/utilities/utils/function_router.py`
   - Follow the naming conventions laid out in this class' docstring
   - This will make the function available in core PresQT code

## 6.4 Resource Upload Endpoint

1. Update your target in `presqt/specs/targets.json` by setting `supported_actions.resource_upload` to `true`.

2. Add a function to perform the resource upload inside of your target directory.
   - If you would like to keep your file/function names consistent with what already exists add this function at `presqt/targets/<target_name>/functions/upload/<target_name>_upload_resource()`
   - The function must have the following parameters **in this order**:

| | | |
|---|---|---|
| token | str | User's token for the target |
| resource_id | str | ID of the resource requested |
| re-source_main_dir | str | Path to the main directory on the server for the resources to be uploaded |
| hash_algorithm | str | Hash algorithm we are using to check for fixity |
| file_duplicate_action | str | The action to take when a duplicate file is found Options: [ignore, update] |
| pro-cess_info_path | str | The path to this download's process_info_path |
| action | str | The type of action occurring |

   - The function must return a **dictionary** with the following keys:

| | | |
|---|---|---|
| resources_ignored | array | Array of string paths of files that were ignored when uploading the resource<br>Path should have the same base as resource_main_dir |
| resources_updated | array | Array of string paths of files that were updated when uploading the resource<br>Path should have the same base as resource_main_dir |
| file_metadata_list | list | List of dictionaries that contains FTS metadata and hash info for each file |
| action_metadata | dict | Dictionary containing FTS metadata about the action occurring |
| project_id | str | ID of the parent project for this upload. Needed for metadata upload |
| project_link | str | The link to either the resource or the home page of the user if not available through API |

**Metadata Dictionary Details**

| | | |
|---|---|---|
| actionRootPath | str | Original path of the file on the server before upload.<br>This is used to connect this metadata with download metadata if the action is a transfer. |
| destinationHash | dict | Hash of the resource in the target that was calculated using the hash_algorithm given as a function parameter<br>Key must be the hash algorithm used value must be the hash itself<br>Can be an empty dict if no hashes exist |
| destinationPath | str | Full path of the file at the destination<br>Start the path with a / |
| title | str | Title of the file |

**Action Metadata Dictionary Details**

| | | |
|---|---|---|
| destinationUsername | str | Username of the user making the request at the destination target |

**Example Resource Upload Function:**

```
def <your_target_name>_upload_resource(token, resource_id,␣
↪resource_main_dir,
                            hash_algorithm, file_duplicate_action):
    # Process to upload resource goes here.
    # Variables below are defined here to show examples of␣
↪structure.
    file_metadata_list = [
        {
            "actionRootPath": 'resource_main_dir/path/to/
↪updated/file.jpg',
```

(continues on next page)

(continued from previous page)

```
            "destinationPath": '/path/to/updated/file.jpg',
            "title": 'file.jpg,
            "destinationHash": {'md5': '123456'} # hash_
↪algorithm = 'md5'
        }
    ]
    resources_ignored = ['path/to/ignored/file.png', 'another/
↪ignored/file.jpg']
    resources_updated = ['path/to/updated/file.jpg']
    action_metadata = {"destinationUsername": 'destination_
↪username'}

    return {
        'resources_ignored': resources_ignored,
        'resources_updated': resources_updated,
        'action_metadata': action_metadata,
        'file_metadata_list': file_metadata_list,
        'project_id': '1234',
        'project_link': 'https://osf.io/1234'
    }
```

3. Add a function to upload FTS metadata to the correct location within the resource's parent project.

   - If you would like to keep your file/function names consistent with what already exists add this function at `presqt/targets/<target_name>/functions/upload_metadata/<target_name>_upload_metadata()`
   - The function must have the following parameters **in this order**:

| | | |
|---|---|---|
| token | str | User's token for the target |
| project_id | str | The id of the parent project for the resource uploaded |
| meta-data_dict | dict | The FTS metadata dictionary to upload<br>At this point it will be a Python dict |

   - The function doesn't return anything

   **Example Resource Upload Function:**

```
def <your_target_name>_upload_metadata(token, project_id,␣
↪metadata_dict):
    # Process to upload metadata goes here.

    # If you want to upload the extra metadata to fields␣
↪supported by your API
    # you will have to add that functionality as well. The␣
↪extra valuees are stored
    # in metadata_dict['extra_metadata]. IE:
    update_project_with_metadata(url, metadata_dict['extra_
↪metadata'])
```

3. Add the resource upload and upload metadata functions to `presqt/api_v1/utilities/utils/function_router.py`

   - Follow the naming conventions laid out in this class' docstring
   - This will make the function available in core PresQT code

## 6.5 Resource Transfer Endpoint

1. Update your target in `presqt/specs/targets.json` by setting `supported_actions.resource_transfer_in`, `supported_actions.resource_transfer_out`, `supported_actions.supported_transfer_partners.transfer_in`, and `supported_actions.supported_transfer_partners.transfer_out` appropriately.

The resource transfer endpoint utilizes the Download and Upload functions. If these two functions are in place then transfer is available.

2. To support `Keyword Enhancement` during the transfer process, add keyword functions as outlined below in the Keyword Enhancement Endpoint section

## 6.6 Keyword Enhancement Endpoint

Targets that want the ability to suggest or enhance new keywords must provide keyword functions.

### 6.6.1 Suggest Keywords

To support the suggestion of `keywords`, a keyword fetch function must be written that will fetch keywords from the target.

1. Update your target in `presqt/specs/targets.json` by setting `keywords` to `true`.
2. Add a function to return a dictionary of keywords found in the target.
   - If you would like to keep your file/function names consistent with what already exists add this function at `presqt/targets/<target_name>/functions/keywords/<target_name>_fetch_keywords()`
   - The function must have the following parameters **in this order**:

| | | |
|---|---|---|
| token | str | User's token for the target |
| resource_id | str | ID for the resource we want to get keywords from |

   - The function must return a dictionary with the following keys:

| | | |
|---|---|---|
| key-words | ar-ray | Array of keywords found in the target |
| <at-tribute_name> | ar-ray | Array of keywords found for this attribute<br>Name the key whatever the attribute name is. See example for more details. |

**Example Keyword Fetch Function:**

```
def <your_target_name>_fetch_keywords(token, resource_id):
    # Process to fetch keywords goes here.
    # Variables below are defined here to show examples of␣
↪structures.
    # This target has keywords in two attributes, 'topics' and
↪'tags'.
    keyword_dictionary = {
```

(continues on next page)

(continued from previous page)

```
        'topics': ['cat', 'dog'],
        'tags': ['food', 'water'],
        'keywords': ['cat', 'dog', 'food', 'water']
    }

    return keyword_dictionary
```

3. Add the keyword fetch function to `presqt/api_v/utilities/utils/function_router.py`
   - Follow the naming conventions laid out in this class' docstring
   - This will make the function available in core PresQT code

### 6.6.2 Enhance Keywords

To support the enhancement of `keywords`, a keyword upload function must be written that will upload new enhanced keywords to the target.

1. Update your target in `presqt/specs/targets.json` by setting `keywords_upload` to `true`.
2. Add a function to upload give keywords to the target.
   - If you would like to keep your file/function names consistent with what already exists add this function to `presqt/targets/<target_name>/functions/keywords/<target_name>_upload_keywords()`
   - The function must have the following parameters **in this order**:

| | | |
|---|---|---|
| token | str | User's token for the target |
| resource_id | str | ID for the resource we want to upload keywords to |
| keywords | list | List of new keywords to upload |

   - The function must return a dictionary with the following keys:

| | | |
|---|---|---|
| updated_keywords | list | List of the final keyword list at the target |
| project_id | str | The ID of the project containing this resource |

**Example Keyword Upload Function:**

```
def <your_target_name>_upload_keywords(token, resource_id, keywords):
    # Process to upload keywords goes here.
    # Variables below are defined here to show examples of structures.
    updated_keywords = ['cat', 'food', 'feline', 'grub']
    project_id = '1234'

    return {'updated_keywords': updated_keywords, 'project_id': project_
↪id}
```

3. Add the keyword upload function to `presqt/api_v/utilities/utils/function_router.py`
   - Follow the naming conventions laid out in this class' docstring
   - This will make the function available in core PresQT code

## 6.7 Error Handling

When any of these target functions are called within PresQT core code they are wrapped inside of a `Try-Except` clause which looks for the exception `PresQTResponseException`. The definition of this exception can be found at `presqt.utilities.exceptions.exceptions.PresQTResponseException`.

CHAPTER

SEVEN

# API ENDPOINTS

## 7.1 Authentication

Refer to the authentication details *here*.

## 7.2 Duplicate File Handling

When `Uploading` or `Transferring` resources a header, `presqt-file-duplicate-action`, must be included. The options are `ignore` or `update`. This header tells the target uploading the resource what to do when a file being uploaded already exists in the source target.

`Ignore` will not update the duplicate file, even if the contents of the files don't match.

`Update` will only update the duplicate file if the contents of the files don't match.

## 7.3 Searching Resource Collections

Search results are ordered by date modified unless the target does not support it.

Only a single search filter can be used at a time.

### 7.3.1 Search Filters

General search across all available target search parameters: `resources/?general=search_value`

Search by project 'title': `resources/?title=Project+Title`

Search by project 'id': `resources/?id=123456`

Search by project 'author': `resources/?author=bfox6`

Search by project 'keywords': `resources/?keywords=cat`

## 7.4 Paginating Resource Collections

Pagination has been added at the collection level to improve load times. Targets now return Pagination information for users resources, as well as searched resources.

### 7.4.1 Page Parameter

Pagination across all available targets: `resources/?page=page_number`

## 7.5 Target Endpoints

### 7.5.1 Target Collection

**GET /api_v1/targets/**

Retrieve details of all `Targets`.

**Example request**:

```
GET /api_v1/targets/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response**:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "name": "osf",
        "readable_name": "OSF",
        "status_url": "https://api.osf.io/v2/nodes/",
        "token_url": "https://osf.io/settings/tokens",
        "supported_actions": {
            "resource_collection": true,
            "resource_detail": true,
            "resource_download": true,
            "resource_upload": true,
            "resource_transfer_in": true,
            "resource_transfer_out": true
            "keywords": true,
            "keywords_upload": true,
        },
        "supported_transfer_partners": {
            "transfer_in": [
                "github",
                "curate_nd"
            ],
            "transfer_out": [
                "github"
            ]
        },
        "supported_hash_algorithms": [
            "sha256",
            "md5"
        ],
        "infinite_depth": true
        "links": [
            {
                "name": "Detail",
```

(continues on next page)

(continued from previous page)

```
                "link": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/",
                "method": "GET"
            }
        ]
    },
    {
        "name": "curate_nd",
        "readable_name": "CurateND",
        "status_url": "https://curate.nd.edu/api/items",
        "token_url": "https://curate.nd.edu/api/access_tokens",
        "supported_actions": {
            "resource_collection": true,
            "resource_detail": true,
            "resource_download": true,
            "resource_upload": false,
            "resource_transfer_in": false,
            "resource_transfer_out": true,
            "keywords": true,
            "keywords_upload": false,
        },
        "supported_transfer_partners": {
            "transfer_in": [],
            "transfer_out": [
                "osf",
                "github"
            ]
        },
        "supported_hash_algorithms": [
            "md5"
        ],
        "infinite_depth": false
        "links": [
            {
                "name": "Detail",
                "link": "https://presqt-prod.crc.nd.edu/api_v1/targets/curate_nd/
↪",
                "method": "GET"
            }
        ]
    }
]
```

**Status Codes**

- 200 OK – `Targets` successfully retrieved

### 7.5.2 Target Details

**GET /api_v1/targets/(str: target_name)/**

Retrieve details of a single `Target`.

**Example request**:

```
GET /api_v1/targets/OSF/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "name": "osf",
    "readable_name": "OSF",
    "status_url": "https://api.osf.io/v2/nodes/",
    "token_url": "https://osf.io/settings/tokens",
    "supported_actions": {
        "resource_collection": true,
        "resource_detail": true,
        "resource_download": true,
        "resource_upload": true,
        "resource_transfer_in": true,
        "resource_transfer_out": true,
        "keywords": true,
        "keywords_upload": true,
    },
    "supported_transfer_partners": {
        "transfer_in": [
            "github",
            "curate_nd"
        ],
        "transfer_out": [
            "github"
        ]
    },
    "supported_hash_algorithms": [
        "sha256",
        "md5"
    ],
    "infinite_depth": true
    "links": [
        {
            "name": "Collection",
            "link": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/resources/
↪",
            "method": "GET"
        },
        {
            "name": "Upload",
            "link": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/resources/
↪",
            "method": "POST"
        },
        {
            "name": "Transfer",
            "link": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/resources/
↪",
            "method": "POST"
        }
    ]
}
```

**Status Codes**

- 200 OK – `Target` successfully retrieved

- 404 Not Found – Invalid `Target` name

## 7.6 Resource Endpoints

### 7.6.1 Resource Collection

**GET /api_v1/targets/(str: target_name)/resources/**

Retrieve details of all top level resources for a given `Target` and `User Token`

**Example request**:

```
GET /api_v1/targets/OSF/resources/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "resources": [
        {
            "kind": "container",
            "kind_name": "project",
            "id": "cmn5z",
            "container": null,
            "title": "Test Project",
            "links": [
                {
                    "name": "Detail",
                    "link": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/
↪resources/cmn5z/",
                    "method": "GET"
                }
            ]
        },
        {
            "kind": "container",
            "kind_name": "project",
            "id": "12345",
            "container": null,
            "title": "Egg Project",
            "links": [
                {
                    "name": "Detail",
                    "link": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/
↪resources/12345/",
                    "method": "GET"
                }
            ]
        }
    ],
    "pages": {
        "first_page": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/
↪resources?page=1",
```

(continues on next page)

(continued from previous page)

```
        "previous_page": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/
↪resources?page=5",
        "next_page": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/resources?
↪page=7",
        "last_page": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/resources?
↪page=30",
        "total_pages": 1,
        "per_page": 10,
        "base_page": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/resources?
↪page="
    }
}
```

**Example request w/ search parameter**:

```
GET /api_v1/targets/OSF/resources?title=egg/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example request w/ search parameter and page parameter**:

```
GET /api_v1/targets/OSF/resources?title=egg&page=3/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

Search filtering rules can be found here.

**Request Headers**

- *presqt-source-token* – User's token for the source target

**Status Codes**

- 200 OK – `Resources` successfully retrieved
- 400 Bad Request – The `Target` does not support the action `resource_collection`
- 400 Bad Request – `presqt-source-token` missing in the request headers
- 400 Bad Request – The `search query` is not formatted correctly.
- 401 Unauthorized – `Token` is invalid
- 404 Not Found – Invalid `Target` name

### 7.6.2 Resource Detail

**GET /api_v1/targets/(str: target_name)/resources/(str: resource_id).json/**

Retrieve details of a `Resource` in JSON format

**Example request**:

```
GET /api_v1/targets/OSF/resources/1234.json/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "kind": "container",
    "kind_name": "project",
    "id": "cmn5z",
    "title": "Test Project",
    "date_created": "2019-05-13T15:06:34.521000Z",
    "date_modified": "2019-05-13T15:06:34.521000Z",
    "hashes": {
        "md5": null,
        "sha256": null
    },
    "extra": {
        "last_touched": "2019-11-07T17:00:51.680957",
        "materialized_path": "/Test Project",
        "current_version": 1,
        "provider": "googledrive",
        "path": "/Test Project",
        "current_user_can_comment": true,
        "guid": "byz93",
        "checkout": null,
        "tags": [],
        "size": null
    },
    "children": [
        {
            "kind": "container",
            "kind_name": "storage",
            "id": "cmn5z:osfstorage",
            "container": "cmn5z",
            "title": "osfstorage",
            "links": [
                {
                    "name": "Detail",
                    "link": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/
↪resources/cmn5z:osfstorage/",
                    "method": "GET"
                }
            ]
        },
        {
            "kind": "container",
            "kind_name": "folder",
            "id": "5cd9832cf244ec0021e5f245",
            "container": "cmn5z:osfstorage",
            "title": "Images",
            "links": [
                {
                    "name": "Detail",
                    "link": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/
↪resources/5cd9832cf244ec0021e5f245/",
                    "method": "GET"
                }
            ]
        },
```

(continues on next page)

(continued from previous page)

```
        {
            "kind": "item",
            "kind_name": "file",
            "id": "5cd98510f244ec001fe5632f",
            "container": "5cd9832cf244ec0021e5f245",
            "title": "22776439564_7edbed7e10_o.jpg",
            "links": [
                {
                    "name": "Detail",
                    "link": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/
↪resources/5cd98510f244ec001fe5632f/",
                    "method": "GET"
                }
            ]
        }
    ],
    "links": [
        {
            "name": "Download",
            "link": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/resources/
↪cmn5z.zip/",
            "method": "GET"
        },
        {
            "name": "Upload",
            "link": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/resources/
↪cmn5z/",
            "method": "POST"
        },
        {
            "name": "Transfer",
            "link": "https://presqt-prod.crc.nd.edu/api_v1/targets/osf/resources/
↪cmn5z/",
            "method": "POST"
        }
    ],
    "actions": [
        "Transfer"
    ]
}
```

**Request Headers**

- *presqt-source-token* – User's token for the source target

**Status Codes**

- 200 OK – `Resource` successfully retrieved
- 400 Bad Request – The `Target` does not support the action `resource_detail`
- 400 Bad Request – `presqt-source-token` missing in the request headers
- 400 Bad Request – Invalid format given. Must be `json`
- 401 Unauthorized – `Token` is invalid
- 403 Forbidden – User does not have access to this `Resource`
- 404 Not Found – Invalid `Target` name

- 404 Not Found – `Resource` with this `ID` not found for this user
- 410 Gone – `Resource` no longer available

## 7.7 Resource Download Endpoints

### 7.7.1 Download Resource

**GET /api_v1/targets/(str: target_name)/resources/(str: resource_id).zip/**

Retrieve a Resource as a ZIP file. This endpoint begins the download process but does not return the zip file. Rather, it returns a link which can be used to the hit the `Job Status` endpoint to check in on the process.

**Example request**:

```
GET /api_v1/targets/OSF/resources/1234.zip/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response**:

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "message": "The server is processing the request.",
    "download_job_zip": "https://presqt-prod.crc.nd.edu/api_v1/job_status/
↪download.zip/",
    "download_job_json": "https://presqt-prod.crc.nd.edu/api_v1/job_status/
↪download.json/"
}
```

**Request Headers**

- *presqt-source-token* – User's token for the source target

**Status Codes**

- 202 Accepted – `Resource` has begun downloading
- 400 Bad Request – The `Target` does not support the action `resource_download`
- 400 Bad Request – User currently has processes in progress.
- 400 Bad Request – `presqt-source-token` missing in the request headers
- 400 Bad Request – `presqt-email-opt-in` missing in the request headers
- 400 Bad Request – Invalid format given. Must be `zip`
- 404 Not Found – Invalid `Target` name

### 7.7.2 Resource Download Job Status

**GET /api_v1/job_status/download.json/**

Use the `Job Status` endpoint to check in on the `Download Process`. Provide the `presqt-source-token` in the headers.

**Example request**

```
GET /api_v1/job_status/download/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response if the download request is still in progress**:

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "job_percentage": 27,
    "status": "in_progress",
    "status_code": null,
    "message": "Downloading files from OSF..."
}
```

**Example response if the download request finished successfully**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "status_code": "200",
    "message": "Download successful. See PRESQT_FTS_METADATA.json for more␣
↪details.",
    "zip_name": "osf_download_cmn5z.zip",
    "failed_fixity": [
        "/Test Project/googledrive/PresQT Swimlane Activity Diagram 03_21_19 (2).
↪pdf",
        "/Test Project/googledrive/module_responses.csv",
        "/Test Project/googledrive/Google Images/IMG_4740.jpg",
        "/Test Project/googledrive/Character Sheet - Alternative - Print Version.
↪pdf"
    ],
    "job_percentage": 100,
    "status": "finished"
}
```

**Example response if the download failed**:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

{
    "job_percentage": 0,
    "status": "failed",
    "status_code": 404,
    "message": "Resource with id 'bad_id' not found for this user."
}
```

**Request Headers**

- *presqt-source-token* – User's `Token` for the source target

**Status Codes**

- 200 OK – `Download` has finished successfully
- 202 Accepted – `Download` is being processed on the server

- 400 Bad Request – `presqt-source-token` missing in the request headers
- 400 Bad Request – Invalid format given. Must be json or zip.
- 404 Not Found – Invalid `Ticket Number`
- 500 Internal Server Error – `Download` failed on the server

**GET /api_v1/job_status/download.zip/**

Check on the `Download Process` for the given user. If download has failed or is in progress this endpoint will return a JSON payload detailing this. If download has completed this endpoint will return the zip file of the resource originally requested.

**Example request**:

```
GET /api_v1/job_status/download.zip/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response if the download request is still in progress**:

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "job_percentage": 27,
    "status": "in_progress",
    "status_code": null,
    "message": "Downloading files from OSF..."
}
```

**Example response if download finished successfully**:

```
HTTP/1.1 200 OK
Content-Type: application/zip

Payload is ZIP file
```

**Example response if the download failed**:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

{
    "job_percentage": 0,
    "status": "failed",
    "status_code": 404,
    "message": "Resource with id 'bad_id' not found for this user."
}
```

**Request Headers**

- *presqt-source-token* – User's `Token` for the source target

**Status Codes**

- 200 OK – `Download` has finished successfully
- 202 Accepted – `Download` is being processed on the server
- 400 Bad Request – `presqt-source-token` missing in the request headers

- 400 Bad Request – Invalid format given. Must be json or zip.
- 404 Not Found – Invalid `Ticket Number`
- 500 Internal Server Error – `Download` failed on the server

**PATCH /api_v1/job_status/upload/**

Cancel the `Download Process` for the given user.‘.

If the download has finished before it can be cancelled it will return the finished info from process_info.json.

If the download was successfully cancelled then it will return the cancelled info from process_info.json.

**Example request**:

```
PATCH /api_v1/job_status/download/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response if download cancelled successfully**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "status_code": "499",
    "message": "Download was cancelled by the user"
}
```

**Example response if download finished before endpoint was able to cancel**:

```
HTTP/1.1 406 OK
Content-Type: application/json

{
    "status_code": "200",
    "message": "Download successful."
}
```

**Request Headers**

- *presqt-source-token* – User’s `Token` for the source target

**Status Codes**

- 200 OK – `Download` cancelled
- 406 Not Acceptable – `Download` finished before cancellation
- 400 Bad Request – `presqt-source-token` missing in the request headers
- 404 Not Found – Invalid `Ticket Number`

## 7.8 Resource Upload Endpoints

### 7.8.1 Upload New Top Level Resource

**POST /api_v1/targets/(str: target_name)/resources/**

Upload a new top level resource, for instance a Project. This endpoint begins the `Upload` process. It returns a link which can be used to the hit the `Job Status` endpoint to check in on the process.

**Example request**:

```
POST /api_v1/targets/OSF/resources/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response**:

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "message": "The server is processing the request.",
    "upload_job": "https://presqt-prod.crc.nd.edu/api_v1/job_status/upload/"
}
```

**Request Headers**

- *presqt-destination-token* – User's `Token` for the destination target
- *presqt-file-duplicate-action* – Action to be taken if a duplicate file is found (Either `update` or `ignore`)

**Form Parameters**

- **`presqt-file`** – The `Resource` to `Upload`. Must be a BagIt file in ZIP format.

**Status Codes**

- 202 Accepted – `Resource` has begun uploading
- 400 Bad Request – The `Target` does not support the action `resource_upload`
- 400 Bad Request – `presqt-destination-token` missing in the request headers
- 400 Bad Request – The file, `presqt-file`, is not found in the body of the request
- 400 Bad Request – The file provided is not a zip file
- 400 Bad Request – The file provided is not in BagIt format
- 400 Bad Request – Checksums failed to validate
- 400 Bad Request – `presqt-file-duplicate-action` missing in the request headers
- 400 Bad Request – `presqt-email-opt-in` missing in the request headers
- 400 Bad Request – Invalid `file_duplicate_action` header give. The options are `ignore` or `update`
- 400 Bad Request – Repository is not formatted correctly. Multiple directories exist at the top level
- 400 Bad Request – Repository is not formatted correctly. Files exist at the top level
- 400 Bad Request – User currently has processes in progress.
- 401 Unauthorized – `Token` is invalid
- 404 Not Found – Invalid `Target` name

### 7.8.2 Upload To Existing Resource

**POST /api_v1/targets/(str: target_name)/resources/(str: resource_id)/**

Upload a resource to an existing container. This endpoint begins the `Upload` process. It returns a link which can be used to the hit the `Job Status` endpoint to check in on the process.

**Example request**:

```
POST /api_v1/targets/OSF/resources/1234/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response**:

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "message": "The server is processing the request.",
    "upload_job": "https://presqt-prod.crc.nd.edu/api_v1/job_status/upload/"
}
```

**Request Headers**

- *presqt-destination-token* – User's `Token` for the destination target
- *presqt-file-duplicate-action* – Action to be taken if a duplicate file is found (Either `update` or `ignore`)

**Form Parameters**

- **`presqt-file`** – The `Resource` to `Upload`. Must be a BagIt file in ZIP format.

**Status Codes**

- 202 Accepted – `Resource` has begun uploading
- 400 Bad Request – The `Target` does not support the action `resource_upload`
- 400 Bad Request – `presqt-destination-token` missing in the request headers
- 400 Bad Request – `presqt-email-opt-in` missing in the request headers
- 400 Bad Request – The file, `presqt-file`, is not found in the body of the request
- 400 Bad Request – The file provided is not a zip file
- 400 Bad Request – The file provided is not in BagIt format
- 400 Bad Request – Checksums failed to validate
- 400 Bad Request – `presqt-file-duplicate-action` missing in the request headers
- 400 Bad Request – Invalid `file_duplicate_action` header give. The options are `ignore` or `update`
- 400 Bad Request – User currently has processes in progress.
- 401 Unauthorized – `Token` is invalid
- 403 Forbidden – User does not have access to this `Resource`
- 404 Not Found – Invalid `Target` name

- 410 Gone – `Resource` no longer available

### 7.8.3 Resource Upload Job Status

**GET /api_v1/job_status/upload/**

Check on the `Upload Process` for the given user.

**Example request**:

```
GET /api_v1/job_status/upload/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response if the upload is in progress**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "status_code": null,
    "message": "Uploading files to OSF...",
    "status": "in_progress",
    "job_percentage": 0
}
```

**Example response if upload finished successfully**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "status_code": "200",
    "message": "Upload successful.",
    "status": "finished",
    "failed_fixity": [],
    "resources_ignored": [],
    "resources_updated": [],
    "job_percentage": 99
}
```

**Example response if upload failed**:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

{
    "job_percentage": 0,
    "status": "failed",
    "status_code": 404,
    "message": "Resource with id 'bad_id' not found for this user."
}
```

**Request Headers**

- *presqt-destination-token* – User's `Token` for the destination target

**Status Codes**

- 200 OK – `Upload` has finished successfully
- 202 Accepted – `Upload` is being processed on the server
- 400 Bad Request – `presqt-destination-token` missing in the request headers
- 404 Not Found – Invalid `Ticket Number`
- 500 Internal Server Error – `Upload` failed on the server

**PATCH /api_v1/job_status/upload/**

Cancel the `Upload Process` for the given user. If the upload has finished before it can be cancelled it will return the finished info from process_info.json. If the upload was successfully cancelled then it will return the cancelled info from process_info.json.

**Example request**:

```
PATCH /api_v1/job_status/upload/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response if upload cancelled successfully**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "status_code": "499",
    "message": "Upload was cancelled by the user"
}
```

**Example response if upload finished before endpoint was able to cancel**:

```
HTTP/1.1 406 OK
Content-Type: application/json

{
    "status_code": "200",
    "message": "Upload successful."
}
```

**Request Headers**

- *presqt-destination-token* – User's `Token` for the destination target

**Status Codes**

- 200 OK – `Upload` cancelled
- 406 Not Acceptable – `Upload` finished before cancellation
- 400 Bad Request – `presqt-destination-token` missing in the request headers
- 404 Not Found – Invalid `Ticket Number`

## 7.9 Resource Transfer Endpoints

**Note:** The Upload and Transfer endpoints are the same POST endpoints **except** the specification of where the source resource is coming from.

For `Uploads` the resource will be a file provided as form-data

For `Transfers` the location of resource (source_target and resource_id) will be specified in the body as JSON

### 7.9.1 Transfer New Top Level Resource

**POST /api_v1/targets/(str: target_name)/resources/**

Transfer a resource from a source target to a destination target. Make the resource a new top level resource, for instance a Project. This endpoint begins the `Transfer` process. It returns a link which can be used to the hit the `Job Status` endpoint to check in on the process.

**Example request**:

```
POST /api_v1/targets/OSF/resources/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json

Example body json:
    {
        "source_target_name":"github",
        "source_resource_id": "209372336",
        "keywords": ["keywords", "to", "add"]
    }
```

**Example response**:

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "message": "The server is processing the request.",
    "transfer_job": "https://presqt-prod.crc.nd.edu/api_v1/job_status/transfer/"
}
```

**Request Headers**

- *presqt-destination-token* – User's `Token` for the destination target
- *presqt-source-token* – User's `Token` for the source target
- *presqt-file-duplicate-action* – Action to be taken if a duplicate file is found (Either `update` or `ignore`)
- *presqt-keyword-action* – Type of keyword action to perform (Either `automatic`, `manual` or `none`)

**JSON Parameters**

- **source_target_name** (*string*) – The `Source Target` where the `Resource` being `Transferred` exists
- **source_resource_id** (*string*) – The ID of the `Resource` to `Transfer`

**Status Codes**

- 202 Accepted – `Resource` has begun transferring
- 400 Bad Request – The `Source Target` does not support the action `resource_transfer_out`

- 400 Bad Request – The Destination Target does not support the action resource_transfer_in
- 400 Bad Request – presqt-source-token missing in the request headers
- 400 Bad Request – presqt-destination-token missing in the request headers
- 400 Bad Request – presqt-file-duplicate-action missing in the request headers
- 400 Bad Request – presqt-email-opt-in missing in the request headers
- 400 Bad Request – Invalid file-duplicate-action header give. The options are ignore or update
- 400 Bad Request – source_resource_id can't be none or blank
- 400 Bad Request – source_resource_id was not found in the request body
- 400 Bad Request – source_target_name was not found in the request body
- 400 Bad Request – keywords was not found in the request body.
- 400 Bad Request – keywords must be in list format.
- 400 Bad Request – Source target does not allow transfer to the destination target
- 400 Bad Request – Destination target does not allow transfer to the source target
- 400 Bad Request – Invalid presqt-keyword-action header given. The options are automatic, manual, or none
- 400 Bad Request – presqt-keyword-action missing in the request headers
- 400 Bad Request – User currently has processes in progress.
- 401 Unauthorized – Source Token is invalid
- 401 Unauthorized – Destination Token is invalid
- 403 Forbidden – User does not have access to the Resource to transfer
- 404 Not Found – Invalid Source Target name
- 404 Not Found – Invalid Destination Target name
- 410 Gone – Resource to transfer is no longer available

## 7.9.2 Transfer To Existing Resource

**POST /api_v1/targets/(str: target_name)/resources/(str: resource_id)/**

Transfer a resource from a source target to a destination target. Transfer to an existing resource. This endpoint begins the Transfer process. It returns a link which can be used to the hit the Job Status endpoint to check in on the process.

**Example request**:

```
POST /api_v1/targets/OSF/resources/1234/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json

Example body json:
    {
        "source_target_name":"github",
```

(continues on next page)

(continued from previous page)

```
        "source_resource_id": "209372336",
        "keywords": ["keywords", "to", "add"]
    }
```

**Example response**:

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "message": "The server is processing the request.",
    "transfer_job": "https://presqt-prod.crc.nd.edu/api_v1/job_status/transfer/"
}
```

**Request Headers**

- *presqt-destination-token* – User's `Token` for the destination target
- *presqt-source-token* – User's `Token` for the source target
- *presqt-file-duplicate-action* – Action to be taken if a duplicate file is found (Either `update` or `ignore`)
- *presqt-keyword-action* – Type of keyword action to perform (Either `automatic`, `manual`, or `none`)

**JSON Parameters**

- **`source_target_name`** (*`string`*) – The `Source Target` where the `Resource` being `Transferred` exists
- **`source_resource_id`** (*`string`*) – The ID of the `Resource` to `Transfer`

**Status Codes**

- 202 Accepted – `Resource` has begun transferring
- 400 Bad Request – The `Source Target` does not support the action `resource_transfer_out`
- 400 Bad Request – The `Destination Target` does not support the action `resource_transfer_in`
- 400 Bad Request – `presqt-source-token` missing in the request headers
- 400 Bad Request – `presqt-destination-token` missing in the request headers
- 400 Bad Request – `presqt-file-duplicate-action` missing in the request headers
- 400 Bad Request – `presqt-email-opt-in` missing in the request headers
- 400 Bad Request – Invalid `file_duplicate_action` header give. The options are `ignore` or `update`
- 400 Bad Request – `source_resource_id` can't be none or blank
- 400 Bad Request – `source_resource_id` was not found in the request body
- 400 Bad Request – `source_target_name` was not found in the request body
- 400 Bad Request – `keywords` was not found in the request body.
- 400 Bad Request – `keywords` must be in list format.

- 400 Bad Request – Source target does not allow transfer to the destination target
- 400 Bad Request – Destination target does not allow transfer to the source target
- 400 Bad Request – Invalid `presqt-keyword-action` header given. The options are `automatic`, `manual` or `none`
- 400 Bad Request – `presqt-keyword-action` missing in the request headers
- 400 Bad Request – User currently has processes in progress.
- 401 Unauthorized – `Source Token` is invalid
- 401 Unauthorized – `Destination Token` is invalid
- 403 Forbidden – User does not have access to the `Resource` to transfer
- 403 Forbidden – User does not have access to the `Resource` to transfer to
- 404 Not Found – Invalid `Source Target` name
- 404 Not Found – Invalid `Destination Target` name
- 410 Gone – `Resource` to transfer is no longer available
- 410 Gone – `Resource` to transfer to is longer available

### 7.9.3 Resource Transfer Job Status

**GET /api_v1/job_status/transfer/**

Check on the `Transfer Process` for the given user.

**Example request**:

```
GET /api_v1/job_status/transfer/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response if transfer is in progress**:

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "status_code": null,
    "message": "Creating PRESQT_FTS_METADATA...",
    "job_percentage": 50
}
```

**Example response if transfer finished successfully**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "status_code": "200",
    "message": "Transfer successful.",
    "job_percentage": 99,
    "failed_fixity": [
        "/PrivateProject/README.md"
    ],
```

(continues on next page)

(continued from previous page)

```
    "resources_ignored": [],
    "resources_updated": [],
    "enhanced_keywords": [
        "EGG",
        "DISORDERED SOLVENT",
        "Electrostatic Gravity Gradiometer",
        "animal house",
        "aqua",
        "Wasser",
    ],
    "initial_keywords": [
        "animals",
        "eggs",
        "water"
    ],
    "source_resource_id": "209372336",
    "destination_resource_id": "qadt3"
}
```

**Example response if transfer failed**:

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json

{
    "status_code": 404,
    "message": "The resource with id, 20938989898989872336, does not exist for␣
↪this user.",
    "job_percentage": 0,
    "status": "failed"
}
```

**Request Headers**

- *presqt-destination-token* – User's `Token` for the destination target
- *presqt-source-token* – User's `Token` for the source target

**Status Codes**

- 200 OK – `Transfer` has finished successfully
- 202 Accepted – `Transfer` is being processed on the server
- 400 Bad Request – `presqt-destination-token` missing in the request headers
- 400 Bad Request – `presqt-source-token` missing in the request headers
- 404 Not Found – Invalid `Ticket Number`
- 500 Internal Server Error – `Transfer` failed on the server

**`PATCH /api_v1/job_status/transfer/`**

Cancel the `Transfer Process` for the given user. If the transfer has finished before it can be cancelled it will return the finished info from process_info.json. If the transfer was successfully cancelled then it will return the cancelled info from process_info.json.

**Example request**:

```
PATCH /api_v1/job_status/transfer/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response if transfer cancelled successfully**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "status_code": "499",
    "message": "Transfer was cancelled by the user"
}
```

**Example response if transfer finished before endpoint was able to cancel**:

```
HTTP/1.1 406 OK
Content-Type: application/json

{
    "status_code": "200",
    "message": "Transfer successful."
}
```

**Request Headers**

- *presqt-destination-token* – User's `Token` for the destination target
- *presqt-source-token* – User's `Token` for the source target

**Status Codes**

- 200 OK – `Transfer` cancelled
- 406 Not Acceptable – `Transfer` finished before cancellation
- 400 Bad Request – `presqt-destination-token` missing in the request headers
- 400 Bad Request – `presqt-source-token` missing in the request headers
- 404 Not Found – Invalid `Ticket Number`

## 7.10 Keyword Enhancement Endpoints

### 7.10.1 Get a Resource's Keywords And Keyword Enhancements

**GET /api_v1/targets/(str: target_name)/resources/(str: resource_id)/keywords/**

Retrieve a resource's keywords that are both stored in the target and in the PresQT Metadata File (if one exists). Send the keywords to a keyword enhancer. Return both the `Target Keywords` and `Enhanced Keywords` in the payload.

**Example request**:

```
GET /api_v1/targets/OSF/resources/1234/keywords/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "keywords": [
        "eggs",
        "animal",
        "water"
    ],
    "enhanced_keywords": [
        "animals",
        "Animals",
        "EGG",
        "Electrostatic Gravity Gradiometer",
        "water",
        "Water",
        "DISORDERED SOLVENT",
        "aqua",
        "Wasser",
        "dihydrogen oxide",
        "OXYGEN ATOM",
        "oxidane",
    ],
    "all_keywords": [
        "animals",
        "Animals",
        "EGG",
        "Electrostatic Gravity Gradiometer",
        "water",
        "Water",
        "DISORDERED SOLVENT",
        "aqua",
        "Wasser",
        "dihydrogen oxide",
        "OXYGEN ATOM",
        "oxidane",
        "eggs",
        "animal",
        "water"
    ]
}
```

**Request Headers**

- *presqt-source-token* – User's `Token` for the source target

**Status Codes**

- 200 OK – Keywords successfully fetched
- 400 Bad Request – The `Source Target` does not support the action `keywords`
- 400 Bad Request – The `resource type` does not support `keywords`
- 401 Unauthorized – `Source Token` is invalid

### 7.10.2 Upload Keywords to a Resource

**POST /api_v1/targets/(str: target_name)/resources/(str: resource_id)/keywords/**

Take a list of keywords and add them to the Resource's keywords both in the target and in the PresQT FTS Metadata file. The returned payload will contain both the new keywords added and the final full list of keywords in the target.

**Example request**:

```
POST /api_v1/targets/OSF/resources/1234/keywords/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json

Example body json:
    {
        "keywords": ["cat", "water"]
    }
```

**Example response**:

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "keywords_added": [
        "feline",
        "aqua",
        "dihydrogen oxide",
        "DISORDERED SOLVENT",
        "EGG",
        "Electrostatic Gravity Gradiometer",
        "oxidane",
        "OXYGEN ATOM",
        "Wasser",
        "Water"
    ],
    "final_keywords": [
        "feline",
        "aqua",
        "dihydrogen oxide",
        "DISORDERED SOLVENT",
        "EGG",
        "eggs",
        "Electrostatic Gravity Gradiometer",
        "oxidane",
        "OXYGEN ATOM",
        "Wasser",
        "water",
        "Water"
    ]
}
```

**Request Headers**

- *presqt-source-token* – User's `Token` for the source target

**JSON Parameters**

- **`keywords`** (*`array`*) – An array of the `keywords` to upload

**Status Codes**

- 202 Accepted – `Keywords successfully uploaded`
- 400 Bad Request – The `Source Target` does not support the action `keywords`
- 400 Bad Request – The `Source Target` does not support the action `keywords_upload`
- 400 Bad Request – The `resource type` does not support `keywords`
- 400 Bad Request – `keywords` is missing from the request body
- 400 Bad Request – `keywords` must be in list format
- 401 Unauthorized – `Source Token` is invalid

CHAPTER

# EIGHT

# WEB SERVICES

## 8.1 Fixity

### 8.1.1 Tools

- Python Hashlib Library
- BagIt Python Validation

### 8.1.2 PresQT Supported Hash Algorithms

The following is a master list of hash algorithms that are both supported by a target and supported by Python's HashLib library:

- sha256
- md5

Each individual target's supported hash algorithms can be found in presqt/specs/targets.json

### 8.1.3 Resource Download Fixity

Fixity is checked during `Resource Download` by comparing the file hashes provided by the source target with hashes that are generated after files are downloaded on to the server. If the provided hash and the calculated hash match then fixity passes!

The download function will try and find a matching hash algorithm between the source target supported algorithms and algorithms supported by the Python Hashlib library to use when generating hashes for files downloaded to the server. If no hash algorithms match or if the source target does not provide file hashes then `md5` is uses as a default. It also counts this situation as fixity passing since we didn't know what the original hash was.

**Valid Hashes Provided + Fixity Passes Example**:

```
{
    "sha256": "343e249fdb0818a58edcc64663e1eb116843b4e1c4e74790ff331628593c02be",
    "md5": "a4536efb47b26eaf509edfdaca442037"
}
```

will yield

```
{
    "hash_algorithm": "sha256",
    "given_hash": "343e249fdb0818a58edcc64663e1eb116843b4e1c4e74790ff331628593c02be",
    "calculated_hash":
↪"343e249fdb0818a58edcc64663e1eb116843b4e1c4e74790ff331628593c02be",
    "fixity": true
}
```

**Valid Hashes Provided + Fixity Fails Example**:

```
{
    "sha256": "343e249fdb0818a58edcc64663e1eb116843b4e1c4e74790ff331628593c02be",
    "md5": "a4536efb47b26eaf509edfdaca442037"
}
```

will yield

```
{
    "hash_algorithm": "sha256",
    "given_hash": "343e249fdb0818a58edcc64663e1eb116843b4e1c4e74790ff331628593c02be",
    "calculated_hash": "12345678",
    "fixity": false
}
```

**Blank Hashes Provided Example**:

```
{
    "sha256": null,
    "md5": null
}
```

will yield

```
{
    "hash_algorithm": "md5",
    "given_hash": null,
    "calculated_hash":
↪"343e249fdb0818a58edcc64663e1eb116843b4e1c4e74790ff331628593c02be",
    "fixity": true
}
```

**Unknown Hashes Provided Example**:

```
{
    "unknown_hasher": "12345",
    "special_hasher": "1234567"
}
```

will yield

```
{
    "hash_algorithm": "md5",
    "given_hash": null,
    "calculated_hash":
↪"343e249fdb0818a58edcc64663e1eb116843b4e1c4e74790ff331628593c02be",
    "fixity": true
}
```

### 8.1.4 Resource Upload Fixity

During the resource upload process, fixity is checked in two locations. First, when files are saved to the disk from the request. Second, after files are uploaded to the target.

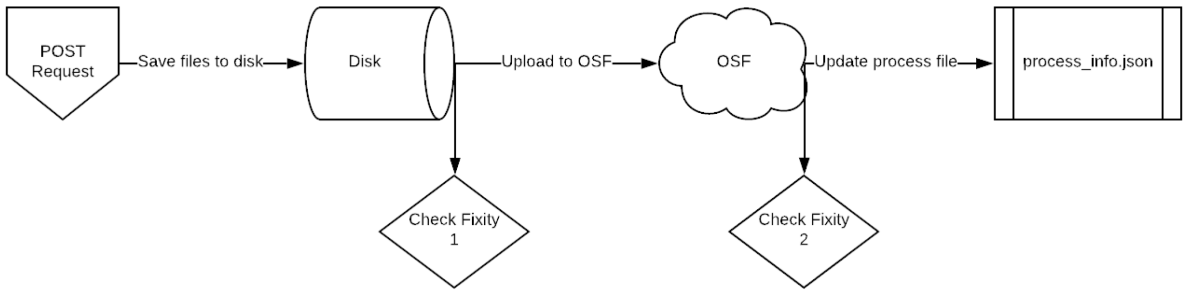


Fig. 1: Image 1: Where in the upload process fixity is checked

#### Fixity Check 1

Resources must be included in the POST request in BagIt format as a zip file. After unzipping the file and saving it to the server we validate the bag using BagIt's built in validator. If any files saved don't match the manifest originally given then the fixity has failed and the server will return an error.

#### Generate New Hashes If Necessary

We now know that the currently saved files are the same as what the user sent forward. Before uploading resources to the target we will make sure that there is a dictionary of hashes available generated by a hash algorithm supported by the target. If the target supports a hash algorithm provided by the resource's 'bag' then we will simply use those. If not, then we need to generate new hashes based on a target supported hash algorithm.

#### Fixity Check 2

After resources are uploaded to the target, we compare the resources' hashes brought back from the target to the hashes we captured before. If any hashes don't match then fixity fails. Since the resources have already been uploaded we simply capture which resources' fixity fails and pass that along the response payload along with the message, 'Upload successful but fixity failed'.

### 8.1.5 Resource Transfer Fixity

Since the `Transfer` endpoint takes advantage of the `Download` and `Upload` endpoints, fixity is checked using all methods already existing in those endpoints.

## 8.2 File Transfer Service (FTS) Metadata

PresQT keeps track of file history of resources being updated by PresQT by passing along an FTS Metadata file with each PresQT action. The file is titled `PRESQT_FTS_METADATA.json`. Every time PresQT takes action on a resource, the source details about the files moved are written to the metadata file.

**Definition of** `PresQT FTS Metadata` **fields**:

| | | |
|---|---|---|
| allKeywords | array | All Keywords added to this resource via PresQT. |
| actions | array | Array of PresQT actions that have taken place on the this project |
| id | string | ID of the PresQT action (uuid4). Created at the time metadata is written |
| actionDateTime | string | Date and time that the action took place |
| actionType | string | Type of action (Download, Upload, Transfer) |
| sourceTargetName | string | Name of the source target the action is taking place on |
| sourceUsername | string | Requesting user's source target username |
| destinationTargetName | string | Name of the destination target the action is taking place on |
| destinationUsername | string | Requesting user's destination target username |
| keywords | dict | Keyword enhancements that took place during this action<br>* Fields found in this dictionaries |
| sourceKeywordsAdded* | array | The source keywords added during this action<br>This includes keywords in the target keywords found in FTS metadata file |
| sourceKeywordsEnhanced* | array | The new keyword enhancements added to the target |
| ontologies | array | Ontologies connected to the enhanced keywords added. |
| enhancer* | str | The enhancement service used to enhance the keywords |
| files | array | Array of files that were involved in the PresQT action |
| sourcePath | string | Path of the file at the source target |
| sourceHashes | dict | Object that contains the file hashes at the source target |
| title | string | Title of the file at the source target |
| extra | dict | Object that contains all extra metadata we can retrieve from the source target |
| failedFixityInfo | array | Array containing dictionaries of info on files that failed fixity check<br>** Fields found in this dictionaries |
| newGeneratedHash** | string | PresQT generated hash of the file |
| algorithmUsed** | string | Hash Algorithm used for the newGeneratedHash |
| reasonFixityFailed** | string | Reason fixity failed for the file |
| destinationPath | string | Path of the file at the destination target |
| destinationHashes | dict | Object that contains the file hashes at the destination target |

**Example of** `PresQT FTS Metadata` **generated by a transfer of a project from GitHub to OSF**:

```
{
    "allKeywords": ["cat", "dog", "feline", "doggo", "pupper"],
    "actions": [
        {
            "id": "bc5a48dc-d1f9-46bd-9137-48fe4843df77",
            "actionDateTime": "2019-11-12 15:45:45.309566+00:00",
            "actionType": "resource_transfer_in",
            "sourceTargetName": "github",
            "sourceUsername": "github_username",
            "destinationTargetName": "osf",
            "destinationUsername": "osf_username",
```

(continues on next page)

(continued from previous page)

```
            "keywords": {
                "sourceKeywordsAdded": ["cat", "dog"],
                "sourceKeywordsEnhanced": ["feline", "doggo", "pupper"],
                "ontologies": [
                    {
                        "keywords": [
                            "doggo",
                            "pupper"
                        ],
                        "ontology": "http://purl.obolibrary.org/obo/CHEBI_153377",
                        "ontology_id": "CHEBI_153377",
                        "categories": [
                            "canine"
                        ]
                    },
                    {
                        "keywords": [
                            "feline"
                        ],
                        "ontology": "http://purl.obolibrary.org/obo/CHEBI_153377",
                        "ontology_id": "CHEBI_153377",
                        "categories": [
                            "felines"
                        ]
                    },
                ],
                "enhancer": "scigraph"
            },
            "files": {
                "created": [
                    {
                        "destinationPath": "NewProject/osfstorage/funnyfunnyimages/
↪Screen_Shot.png",
                        "destinationHashes": {
                            "md5": "3505a89c3cbb82873a107ae41f3997c3"
                        },
                        "failedFixityInfo": [
                            {
                                "NewGeneratedHash": "3505a89c3cbb82873a107ae41f3997c3
↪",

                                "algorithmUsed": "md5",
                                "reasonFixityFailed": "Either a Source Hash was not␣
↪provided or the source hash algorithm is not supported."
                            }
                        ],
                        "title": "Screen_Shot.png",
                        "sourceHashes": {},
                        "sourcePath": "/NewProject/funnyfunnyimages/Screen_Shot.png",
                        "extra": {
                            "commit_hash": "211ef8db83612802aeea151a0e04badfe287bcb9",
                            "size": 731202,
                            "url": "https://api.github.com/repos/presqt-test-user/
↪NewProject/contents/funnyfunnyimages/Screen_Shot.png?ref=master",
                            "html_url": "https://github.com/presqt-test-user/
↪NewProject/blob/master/funnyfunnyimages/Screen_Shot.png",
                            "git_url": "https://api.github.com/repos/presqt-test-user/
↪NewProject/git/blobs/211ef8db83612802aeea151a0e04badfe287bcb9",
```

(continues on next page)

(continued from previous page)

```
                            "download_url": "https://raw.githubusercontent.com/presqt-
↪test-user/NewProject/master/funnyfunnyimages/Screen_Shot.png",
                            "type": "file",
                            "_links": {
                                "self": "https://api.github.com/repos/presqt-test-
↪user/NewProject/contents/funnyfunnyimages/Screen_Shot.png?ref=master",
                                "git": "https://api.github.com/repos/presqt-test-user/
↪NewProject/git/blobs/211ef8db83612802aeea151a0e04badfe287bcb9",
                                "html": "https://github.com/presqt-test-user/
↪NewProject/blob/master/funnyfunnyimages/Screen_Shot.png"
                            }
                        }
                    }
                ],
                "updated": [],
                "ignored": []
            }
        }
    ]
}
```

**Now if we download from OSF the same project that was just transferred, then** `PresQT FTS Metadata` **would be**:

```
{
    "allKeywords": ["cat", "dog", "feline", "doggo", "pupper"],
    "actions": [
        {
            "id": "bc5a48dc-d1f9-46bd-9137-48fe4843df77",
            "actionDateTime": "2019-11-12 15:45:45.309566+00:00",
            "actionType": "resource_transfer_in",
            "sourceTargetName": "github",
            "sourceUsername": "github_username",
            "destinationTargetName": "osf",
            "destinationUsername": "osf_username",
            "keywords": {
                "sourceKeywordsAdded": ["cat", "dog"],
                "sourceKeywordsEnhanced": ["feline", "doggo"],
                "ontologies": [
                    {
                        "keywords": [
                            "doggo",
                            "pupper"
                        ],
                        "ontology": "http://purl.obolibrary.org/obo/CHEBI_153377",
                        "ontology_id": "CHEBI_153377",
                        "categories": [
                            "canine"
                        ]
                    },
                    {
                        "keywords": [
                            "feline"
                        ],
                        "ontology": "http://purl.obolibrary.org/obo/CHEBI_153377",
                        "ontology_id": "CHEBI_153377",
```

(continues on next page)

(continued from previous page)

```
                        "categories": [
                            "felines"
                        ]
                    },
                ],
                "enhancer": "scigraph"
            },
            "files": {
                "created": [
                    {
                        "destinationPath": "NewProject/osfstorage/funnyfunnyimages/
↪Screen_Shot.png",
                        "destinationHashes": {
                            "md5": "3505a89c3cbb82873a107ae41f3997c3"
                        },
                        "failedFixityInfo": [
                            {
                                "NewGeneratedHash": "3505a89c3cbb82873a107ae41f3997c3
↪",
                                "algorithmUsed": "md5",
                                "reasonFixityFailed": "Either a Source Hash was not␣
↪provided or the source hash algorithm is not supported."
                            }
                        ],
                        "title": "Screen_Shot.png",
                        "sourceHashes": {},
                        "sourcePath": "/NewProject/funnyfunnyimages/Screen_Shot",
                        "extra": {
                            "commit_hash": "211ef8db83612802aeea151a0e04badfe287bcb9",
                            "size": 731202,
                            "url": "https://api.github.com/repos/presqt-test-user/
↪NewProject/contents/funnyfunnyimages/Screen_Shot.png?ref=master",
                            "html_url": "https://github.com/presqt-test-user/
↪NewProject/blob/master/funnyfunnyimages/Screen_Shot.png",
                            "git_url": "https://api.github.com/repos/presqt-test-user/
↪NewProject/git/blobs/211ef8db83612802aeea151a0e04badfe287bcb9",
                            "download_url": "https://raw.githubusercontent.com/presqt-
↪test-user/NewProject/master/funnyfunnyimages/Screen_Shot.png",
                            "type": "file",
                            "_links": {
                                "self": "https://api.github.com/repos/presqt-test-
↪user/NewProject/contents/funnyfunnyimages/Screen_Shot.png?ref=master",
                                "git": "https://api.github.com/repos/presqt-test-user/
↪NewProject/git/blobs/211ef8db83612802aeea151a0e04badfe287bcb9",
                                "html": "https://github.com/presqt-test-user/
↪NewProject/blob/master/funnyfunnyimages/Screen_Shot.png"
                            }
                        }
                    }
                ],
                "updated": [],
                "ignored": []
            }
        },
        {
            "id": "bc5a48dc-d1f9-46bd-9137-48fe4843df77",
            "actionDateTime": "2019-11-12 15:45:45.309566+00:00",
```

(continues on next page)

(continued from previous page)

```
            "actionType": "resource_download",
            "sourceTargetName": "osf",
            "sourceUsername": "osf_username",
            "destinationTargetName": "Local Machine",
            "destinationUsername": null,
            "keywords": {},
            "files": {
                "created": [
                    {
                        "destinationPath": "/NewProject/osfstorage/funnyfunnyimages/
↪Screen_Shot.png",
                        "destinationHashes": {},
                        "failedFixityInfo": [],
                        "title": "Screen_Shot.png",
                        "sourceHashes": {
                            "sha256":
↪"6d33275234b28d77348e4e1049f58b95a485a7a441684a9eb9175d01c7f141ea",
                            "md5": "3505a89c3cbb82873a107ae41f3997c3"
                        },
                        "sourcePath": "/NewProject/osfstorage/funnyfunnyimages/Screen_
↪Shot.png",
                        "extra": {
                            "id": "5dcc215848a1d9000cd0a3fb",
                            "parent_project_id": "2bw9j",
                            "endpoint": "https://api.osf.io/v2/files/
↪5dcc215848a1d9000cd0a3fb/",
                            "download_url": "https://files.osf.io/v2/resources/2bw9j/
↪providers/osfstorage/5dcc215848a1d9000cd0a3fb",
                            "upload_url": "https://files.osf.io/v2/resources/2bw9j/
↪providers/osfstorage/5dcc215848a1d9000cd0a3fb",
                            "delete_url": "https://files.osf.io/v2/resources/2bw9j/
↪providers/osfstorage/5dcc215848a1d9000cd0a3fb",
                            "last_touched": null,
                            "date_modified": "2019-11-13T15:29:29.043502Z",
                            "current_version": 1,
                            "date_created": "2019-11-13T15:29:29.043502Z",
                            "provider": "osfstorage",
                            "path": "/5dcc215848a1d9000cd0a3fb",
                            "current_user_can_comment": true,
                            "guid": null,
                            "checkout": null,
                            "tags": [],
                            "size": 731202
                        }
                    }
                ],
                "updated": [],
                "ignored": []
            }
        }
    ]
}
```

### 8.2.1 Metadata Location When Downloading

The `PresQT FTS Metadata` file will be written to the highest level possible of the resource being downloaded.

### 8.2.2 Metadata Location When Uploading or Transferring

The `PresQT FTS Metadata` file will be written to the highest level possible of the destination project. Since this possible level may vary for any target, we leave it up to the target to handle this when they integrate with Upload.

### 8.2.3 Existing Metadata

If a valid `PresQT FTS Metadata` file is found at the top level of the resource being affected by the action then we will add a new action to this existing metadata file.

If an invalid `PresQT FTS Metadata` file is found at the top level of the resource being affected by the action then we will rename the invalid metadata file to `INVALID_PRESQT_FTS_METADATA.json` and then we will create a new valid metadata file with the current actions metadata.

## 8.3 Keyword Assignment

### 8.3.1 Keyword Enhancers

- SciGraph http://ec-scigraph.sdsc.edu:9000/scigraph/docs/

### 8.3.2 Keyword Difference Between Targets

Each target holds keywords in different attributes. Some may have keywords in multiple attributes. The following table outlines the keyword attributes for each target.

| Targets | Keyword Attributes |
|---|---|
| OSF | [Tags] |
| Github | [Topics] |
| Gitlab | [Tag List] |
| CurateND | [Subjects] |
| Zenodo | [Keywords] |
| FigShare | [Tags] |

### 8.3.3 Keyword Assignment During Transfer

When transferring a resource you have the option of either manual or automatic keyword enhancement. Manual enhancement will only add `source` keywords and the keywords provided in the request body. Automatic will add all enhancements including any provided in the request body. These can be set by setting `presqt-keyword-action` in the headers to either `manual` or `automatic`

#### Manual Keywords

If `presqt-keyword-action` is `manual` then PresQT will only add keywords found in the source target and keywords given in the body of the request. This means you need to get the possible enhancements before initiating a transfer.

### Automatic Keywords

If `presqt-keyword-action` is `automatic` then PresQT will add keywords found in the source, keywords given in the request body, and any keyword enhancements found during the transfer process. The following steps occur during the transfer in this case:

1. Fetch all source keywords both in the target and in the FTS metadata file for the transferred resource.
2. Get enhancements with the given enhancer (Defaults to SciGraph for now).
3. Upload keyword enhancements to the `Source Target` and `Destination Target`.
4. Add the keyword enhancements to the FTS Metadata file that gets written to the `Destination Target` during the transfer.
5. Add the keyword enhancements to the FTS Metadata file that gets written to the `Source Target` during the transfer.

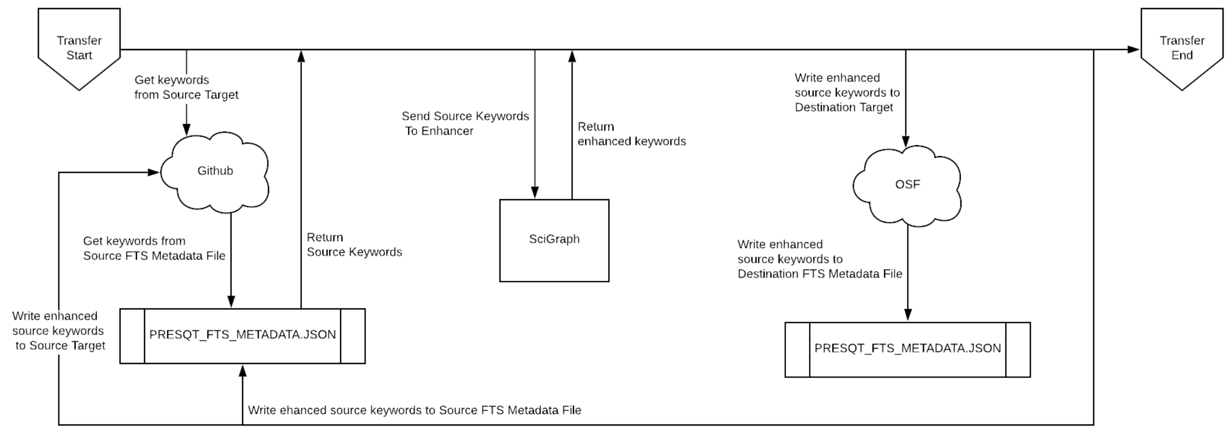

Transfer Start
Get keywords from Source Target
Github
Get keywords from Source FTS Metadata File
Return Source Keywords
Write enhanced source keywords to Source Target
PRESQT_FTS_METADATA.JSON
Write ehanced source keywords to Source FTS Metadata File
Send Source Keywords To Enhancer
Return enhanced keywords
SciGraph
Write enhanced source keywords to Destination Target
OSF
Write enhanced source keywords to Destination FTS Metadata File
PRESQT_FTS_METADATA.JSON
Transfer End

Fig. 2: Image 2: Lifecycle of Keyword Enhancement during a transfer

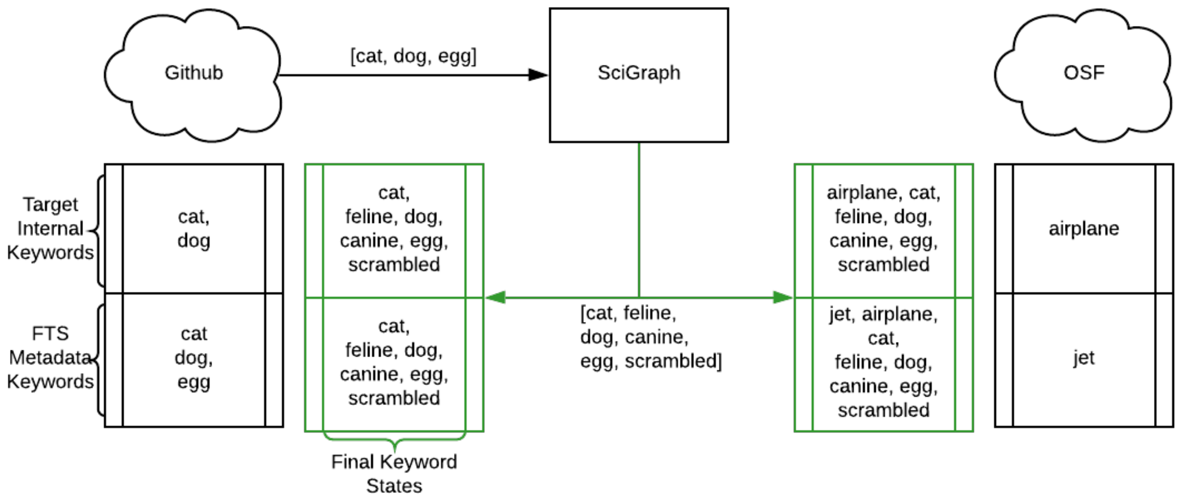

Github
[cat, dog, egg]
SciGraph
OSF
Target Internal Keywords
FTS Metadata Keywords
cat, dog
cat dog, egg
cat, feline, dog, canine, egg, scrambled
cat, feline, dog, canine, egg, scrambled
Final Keyword States
[cat, feline, dog, canine, egg, scrambled]
airplane, cat, feline, dog, canine, egg, scrambled
jet, airplane, cat, feline, dog, canine, egg, scrambled
airplane
jet

Fig. 3: Image 3: Practical Example of Keyword Enhancement during a transfer

### 8.3.4 Keyword Assignment Service Endpoint

Keyword Enhancement can be done without transferring.

1. Use the `Keyword Enhancement GET` endpoint to fetch the keywords from the resource.
2. Pass the keywords you want to enhance to the `Keyword Enhancement POST` endpoint.
3. Enhanced keywords will get uploaded to the target and a new action will get written to the FTS metadata file.

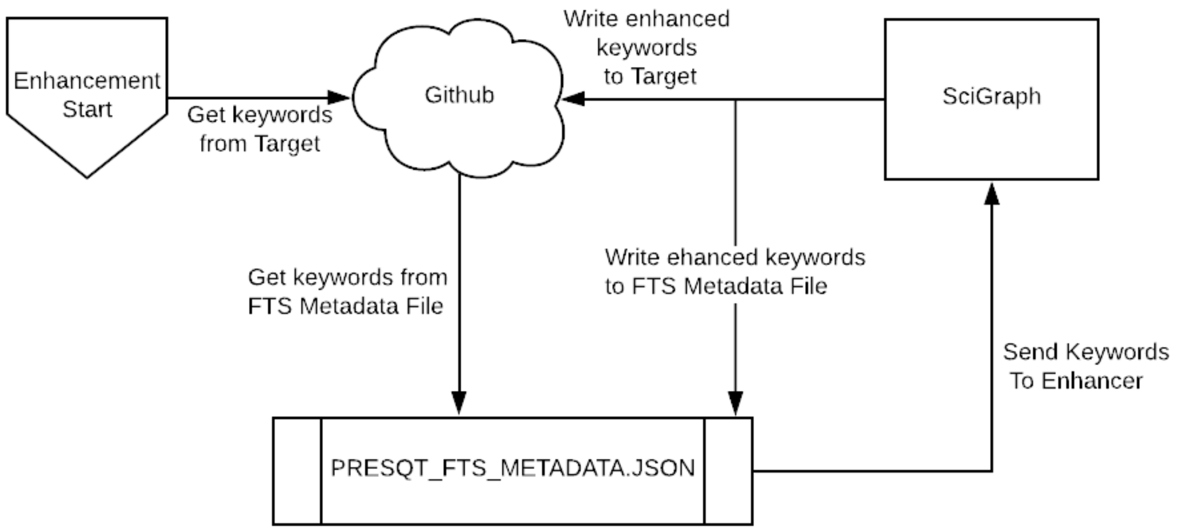

Enhancement Start
Get keywords from Target
Github
Write enhanced keywords to Target
SciGraph
Get keywords from FTS Metadata File
Write ehanced keywords to FTS Metadata File
Send Keywords To Enhancer
PRESQT_FTS_METADATA.JSON

Fig. 4: Image 4: Lifecycle of a Keyword Enhancement Service

## 8.4 Preservation Quality

IN PROGRESS

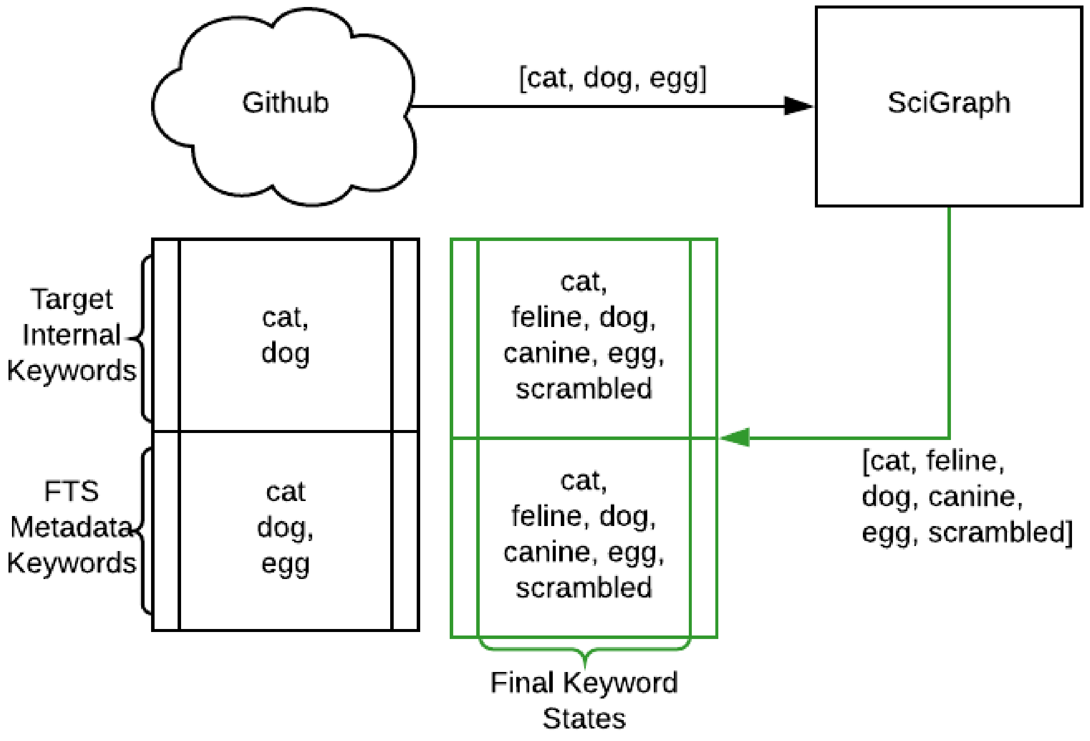

Github
[cat, dog, egg]
SciGraph
Target Internal Keywords
cat, dog
FTS Metadata Keywords
cat dog, egg
cat, feline, dog, canine, egg, scrambled
cat, feline, dog, canine, egg, scrambled
[cat, feline, dog, canine, egg, scrambled]
Final Keyword States

Fig. 5: Image 5: Practical Example of a Keywords Enhancement Service

CHAPTER

NINE

# SERVICES

A service is a unique target integration that is not accessible via the normal API endpoints. Typically, the point of a service is to take action on a resource (emulate, annotate, etc.) rather than simply storing it.

## 9.1 EaaSI (Emulation-as-a-Service Infrastructure) Service

PresQT takes advantage of EaaSI's ability to interpret resources and suggest a relevant emulation environment. Our PresQT API calls use EaaSI's Proposal API to send resources to EaaSI.

### 9.1.1 Step 1: Download the Resource

PresQT is able to use the existing download endpoint to fetch a Target's resource to a PresQT server.

### 9.1.2 Step 2: Start a proposal task on an EaaSI server

Then, using the ticket number created from the PresQT download task, a POST request can be made to PresQT to send EaaSI a url where the downloaded resource can be fetched. During this POST request we write a one time use token to the downloaded resource's process_info.json file. The URL we send to EaaSI to fetch the PresQT resource has a query parameter with this token. This EaaSI download endpoint is for EaaSI use only.

### 9.1.3 Step 3: Get proposal status from EaaSI

We then have a GET endpoint that makes a request to EaaSI to find the progress of the Proposal Task. If the task is complete then we return the url for the suggested emulation environment. Otherwise, we return a 202 status and let the user know the proposal task is still in progress.

## 9.2 FAIR Evaluator Service

PresQT takes advantage of FAIRshare's prebuilt maturity indicator tests. Our PresQT API calls use an approved collection of tests identified by the PI's and community.

## 9.3 FAIRshake Assessment Service

PresQT takes advantage of FAIRshake's manual assessment functionality to allow users to assess the FAIRness of their research projects.

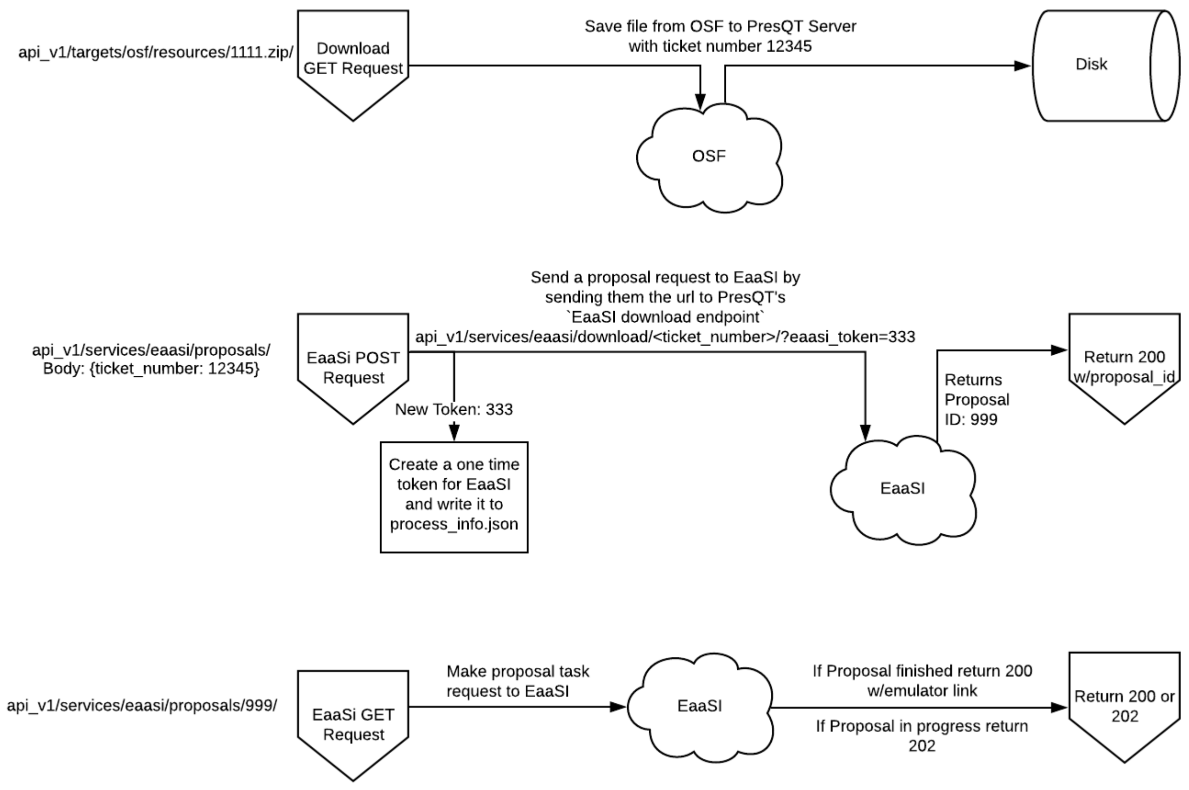

api_v1/targets/osf/resources/1111.zip/
Download GET Request
Save file from OSF to PresQT Server with ticket number 12345
OSF
Disk
api_v1/services/eaasi/proposals/
Body: {ticket_number: 12345}
EaaSi POST Request
Send a proposal request to EaaSI by sending them the url to PresQT's `EaaSI download endpoint` api_v1/services/eaasi/download/<ticket_number>/?eaasi_token=333
New Token: 333
Create a one time token for EaaSI and write it to process_info.json
EaaSI
Returns Proposal ID: 999
Return 200 w/proposal_id
api_v1/services/eaasi/proposals/999/
EaaSi GET Request
Make proposal task request to EaaSI
EaaSI
If Proposal finished return 200 w/emulator link
If Proposal in progress return 202
Return 200 or 202

Fig. 1: Image 1: Workflow of getting an EaaSI Emulation Environment of a given resource

CHAPTER

TEN

# SERVICE ENDPOINTS

## 10.1 Service Endpoints

### 10.1.1 Service Collection

**GET /api_v1/services/**

Retrieve details of all `Services`.

**Example request**:

```
GET /api_v1/services/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response**:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "name": "eaasi",
        "readable_name": "EaaSI",
        "links": [
            {
                "name": "Detail",
                "link": "https://presqt-prod.crc.nd.edu/api_v1/services/eaasi/",
                "method": "GET"
            }
        ]
    }
]
```

**Status Codes**

- 200 OK – `Services successfully retrieved`

### 10.1.2 Service Details

**GET /api_v1/services/(str: service_name)/**

Retrieve details of a single `Service`.

**Example request**:

```
GET /api_v1/services/eaasi/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "name": "eaasi",
    "readable_name": "EaaSI",
    "links": [
        {
            "name": "Proposals",
            "link": "https://presqt-prod.crc.nd.edu/api_v1/services/eaasi/
↪proposals/",
            "method": "POST"
        }
    ]
}
```

**Status Codes**

- 200 OK – `Service` successfully retrieved
- 404 Not Found – Invalid `Service` name

## 10.2 Keyword Enhancement

### 10.2.1 Get Keyword Enhancements From A List Of Keywords

**`GET /api_v1/services/presqt/keyword_enhancement/`**

Take a list of keywords and run them through the keyword enhancement service. The returned payload will contain both the new keywords added and the final full list of keywords.

There are separate endpoints for keyword enhancements through Targets. See the API Endpoint documentation to learn more.

**Example request**:

```
POST /api_v1/services/presqt/keyword_enhancement/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json

Example body json:
    {
        "keywords": ["cat", "water"]
    }
```

**Example response**:

```
HTTP/1.1 202 Accepted
Content-Type: application/json

```

(continues on next page)

(continued from previous page)

```
{
    "keywords_added": [
        "feline",
        "aqua",
        "dihydrogen oxide",
        "DISORDERED SOLVENT",
        "EGG",
        "Electrostatic Gravity Gradiometer",
        "oxidane",
        "OXYGEN ATOM",
        "Wasser",
        "Water"
    ],
    "final_keywords": [
        "feline",
        "aqua",
        "dihydrogen oxide",
        "DISORDERED SOLVENT",
        "EGG",
        "eggs",
        "Electrostatic Gravity Gradiometer",
        "oxidane",
        "OXYGEN ATOM",
        "Wasser",
        "water",
        "Water"
    ]
}
```

**JSON Parameters**

- **keywords** (*array*) – An array of the `keywords` to upload

**Status Codes**

- 202 Accepted – `Keywords successfully uploaded`

## 10.3 EaaSI Endpoints

### 10.3.1 Submit EaaSI Proposal

**POST /api_v1/services/eaasi/proposals/**

Send a file from a PresQT server to start a `proposal task` on an EaaSI server.

**Example request**:

```
POST /api_v1/services/eaasi/proposals/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json

Example body json:
    {
        "ticket_number":"39e56297-04cc-440a-b73e-9788b220f12b"
    }
```

**Example response**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": "19",
    "message": "Proposal task was submitted."
    "proposal_link": "https://presqt-prod.crc.nd.edu/api_v1/services/eaasi/1/"
}
```

**Status Codes**

- 200 OK – Proposal successfully started.
- 400 Bad Request – 'presqt-source-token' missing in request headers
- 400 Bad Request – A download does not exist for this user on the server.
- 404 Not Found – Invalid ticket number
- 404 Not Found – A resource_download does not exist for this user on the server.

### 10.3.2 Get EaaSI Proposal

**GET /api_v1/services/eaasi/proposals/(str: proposal_id)/**
Check on the state of the EaaSI Proposal Task on the EaaSI server.

**Example request**:

```
GET /api_v1/services/eaasi/proposals/12/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response if the proposal task is not finished**:

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "message": "Proposal task is still in progress."
}
```

**Example response if the proposal task is finished successfully**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "image_url": "https://eaasi-portal.emulation.cloud:443/blobstore/api/v1/blobs/
↪imagebuilder-outputs/2ca330d6-23f7-4f0a-943a-e3984b29642c?access_token=default",
    "image_type": "cdrom",
    "environments": [],
    "suggested": {}
}
```

**Status Codes**

- 200 OK – `Proposal Task` has finished successfully

- 202 Accepted – `Proposal Task` is being processed on the EaaSI server
- 404 Not Found – Invalid `Proposal ID`

### 10.3.3 EaaSI Download

**GET /api_v1/services/eaasi/(str: ticket_number)/?eaasi_token=(str: eaasi_token)**
EaaSI specific download endpoint that exposes a resource on a PresQT server to download.

**Example request**:

```
GET /api_v1/services/eeasi/download/39e56297-04cc-440a-b73e/?eaasi=E9luKQU9Ywe5j␣
↪HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response**:

```
HTTP/1.1 200 OK
Content-Type: application/zip

Payload is ZIP file
```

**Status Codes**

- 200 OK – File successfully retrieved.
- 400 Bad Request – `eaasi_token` not found as query parameter.
- 401 Unauthorized – `eaasi_token` does not match the 'eaasi_token' for this server process.
- 404 Not Found – File unavailable.
- 404 Not Found – Invalid ticket number.
- 404 Not Found – A resource_download does not exist for this user on the server.

## 10.4 FAIRshare Endpoints

### 10.4.1 Get FAIRshare Tests

**GET /api_v1/services/fairshare/evaluator/**
Get a list of tests from FAIRshare that are currently supported by PresQT.

**Example request**:

```
GET /api_v1/services/fairshare/evaluator/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response**:

```
HTTP/1.1 200 OK
Content-Type: application/json

```

(continues on next page)

(continued from previous page)

```
[
    {
        "test_name": "FAIR Metrics Gen2- Unique Identifier "
        "description": "Metric to test if the metadata resource has a unique␣
↪identifier. This is done by comparing the GUID to the patterns (by regexp) of␣
↪known GUID schemas such as URLs and DOIs. Known schema are registered in␣
↪FAIRSharing (https://fairsharing.org/standards/?q=&selected_facets=type_
↪exact:identifier%20schema)",
        "test_id": 1
    },
    {
        "test_name": "FAIR Metrics Gen2 - Identifier Persistence "
        "description": "Metric to test if the unique identifier of the metadata␣
↪resource is likely to be persistent. Known schema are registered in FAIRSharing␣
↪(https://fairsharing.org/standards/?q=&selected_facets=type_exact:identifier
↪%20schema). For URLs that don't follow a schema in FAIRSharing we test known␣
↪URL persistence schemas (purl, oclc, fdlp, purlz, w3id, ark).",
        "test_id": 2
    }...
]
```

**Status Codes**

- 200 OK – Tests returned successfully

### 10.4.2 POST FAIRshare Evaluator

**POST /api_v1/services/fairshare/evaluator/**

Submit a FAIRshare Evaluation request with a doi and list of test ids.

**Example request**:

```
POST /api_v1/services/fairshare/evaluator/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json

Example body json:
    {
        "resource_id":"10.17605/OSF.IO/EGGS12",
        "tests": [1, 2]
    }
```

**Example response**:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        "metric_link": "https://w3id.org/FAIR_Evaluator/metrics/1",
        "test_name": "FAIR Metrics Gen2- Unique Identifier ",
        "description": "Metric to test if the metadata resource has a unique␣
↪identifier. This is done by comparing the GUID to the patterns (by regexp) of␣
↪known GUID schemas such as URLs and DOIs. Known schema are registered in␣
↪FAIRSharing (https://fairsharing.org/standards/?q=&selected_facets=type_
↪exact:identifier%20schema)",
```

(continues on next page)

(continued from previous page)

```
        "successes": [
            "Found an identifier of type 'doi'"
        ],
        "failures": [],
        "warnings": []
    },
    {
        "metric_link": "https://w3id.org/FAIR_Evaluator/metrics/2",
        "test_name": "FAIR Metrics Gen2 - Identifier Persistence ",
        "description": "Metric to test if the unique identifier of the metadata␣
↪resource is likely to be persistent. Known schema are registered in FAIRSharing␣
↪(https://fairsharing.org/standards/?q=&selected_facets=type_exact:identifier
↪%20schema). For URLs that don't follow a schema in FAIRSharing we test known␣
↪URL persistence schemas (purl, oclc, fdlp, purlz, w3id, ark).",
        "successes": [
            "The GUID of the metadata is a doi, which is known to be persistent."
        ],
        "failures": [],
        "warnings": []
    }
]
```

**Status Codes**

- 200 OK – Evaluation completed successfully.
- 400 Bad Request – 'resource_id' missing in the request body.
- 400 Bad Request – 'tests' missing in the request body.
- 400 Bad Request – 'tests' must be in list format.
- 400 Bad Request – At least one test is required. Options are: [. . . . . . .]
- 400 Bad Request – 'eggs' not a valid test name. Options are: [. . . . . . .]
- 503 Service Unavailable – FAIRshare returned a <status_code> error trying to process the request

## 10.5 FAIRshake Endpoints

### 10.5.1 Get FAIRshake Rubrics

**GET /api_v1/services/fairshake/rubric/{str: rubric_id}/**

Get a list of merics from FAIRshake that are associated with the rubric id.

**Example request**:

```
GET /api_v1/services/fairshake/rubric/9/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json
```

**Example response**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "metrics": {
        "30": "The structure of the repository permits efficient discovery of␣
↪data and metadata by end users.",
        "31": "The repository uses a standardized protocol to permit access by␣
↪users.",
        "32": "The repository provides contact information for staff to enable␣
↪users with questions or suggestions to interact with repository experts.",
        "33": "Tools that can be used to analyze each dataset are listed on the␣
↪corresponding dataset pages.",
        "34": "The repository maintains licenses to manage data access and use.",
        "35": "The repository hosts data and metadata according to a set of␣
↪defined criteria to ensure that the resources provided are consistent with the␣
↪intent of the repository.",
        "36": "The repository provides documentation for each resource to permit␣
↪its complete and accurate citation.",
        "37": "A description of the methods used to acquire the data is provided.
↪",
        "38": "Version information is provided for each resource, where available.
↪"
    },
    "answer_options": {
        "0.0": "no",
        "0.25": "nobut",
        "0.5": "maybe",
        "0.75": "yesbut",
        "1.0": "yes"
    }
}
```

**Status Codes**

- 200 OK – Rubric returned successfully
- 400 Bad Request – ‘egg’ is not a valid rubric id. Choices are: [‘7’, ‘8’, ‘9’]

### 10.5.2 POST FAIRshake Assessment

**POST /api_v1/services/fairshake/rubric/{str: rubric_id}/**

Submit a FAIRshake Assessment request for the given rubric.

**Example request**:

```
POST /api_v1/services/fairshake/rubric/9/ HTTP/1.1
Host: presqt-prod.crc.nd.edu
Accept: application/json

Example body json:
    {
        "project_url": "https://github.com/ndlib/presqt",
        "project_title": "presqt",
        "rubric_answers": {
            "30": "0.0",
```

(continues on next page)

(continued from previous page)

```
            "31": "0.5",
            "32": "0.0",
            "33": "1.0",
            "34": "1.0",
            "35": "1.0",
            "36": "0.5",
            "37": "0.0",
            "38": "0.0"
        }
    }
```

**Example response**:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "digital_object_id": 166055,
    "rubric_responses": [
        {
            "metric": "The structure of the repository permits efficient␣
↪discovery of data and metadata by end users.",
            "score": "0.0",
            "score_explanation": "no"
        }...
    ]
}
```

**Status Codes**

- 200 OK – Assessment completed successfully.
- 400 Bad Request – ‘eggs’ is not a valid rubric id. Options are: [‘7’, ‘8’, ‘9’]
- 400 Bad Request – ‘project_url’ missing in POST body.
- 400 Bad Request – ‘project_title’ missing in POST body.
- 400 Bad Request – ‘rubric_answers’ missing in POST body.
- 400 Bad Request – ‘rubric_answers’ must be an object with the metric id’s as the keys and answer values as the values.
- 400 Bad Request – Missing response for metric ‘30’. Required metrics are: [‘30’, ‘31’, ‘32’]
- 400 Bad Request – ‘egg’ is not a valid answer. Options are: [‘0.0’, ‘0.25’, ‘0.5’, ‘0.75’, ‘1.0’]
- 400 Bad Request – ‘egg’ is not a valid metric. Required metrics are: [‘30’, ‘31’, ‘32’]

CHAPTER

# ELEVEN

# RESOURCES

This page contains all relevant resources used during development

## 11.1 Links

-
-
-
-
-
-

## 11.2 Example BagIts

### 11.2.1 BagIt Zip files

Since the upload endpoint requires a BagIt file in zip format here are some pre-made zip files to test the upload endpoint.

`#1 Valid BagIt For Top Level Container w/Folder`

`#2 Valid BagIt For Top Level Container w/File`

`#3 Valid BagIt For Existing Container w/Single File`

`#4 Valid BagIt For Existing Container w/Folders & Files`

`#5 Invalid BagIt - Bad Manifest`

`#6 Invalid BagIt - Missing File`

`#7 Invalid BagIt - Unknown File`

### 11.2.2 Example Workflow

The following are instructions on how the BagIt files above can be used to test the Upload endpoint:

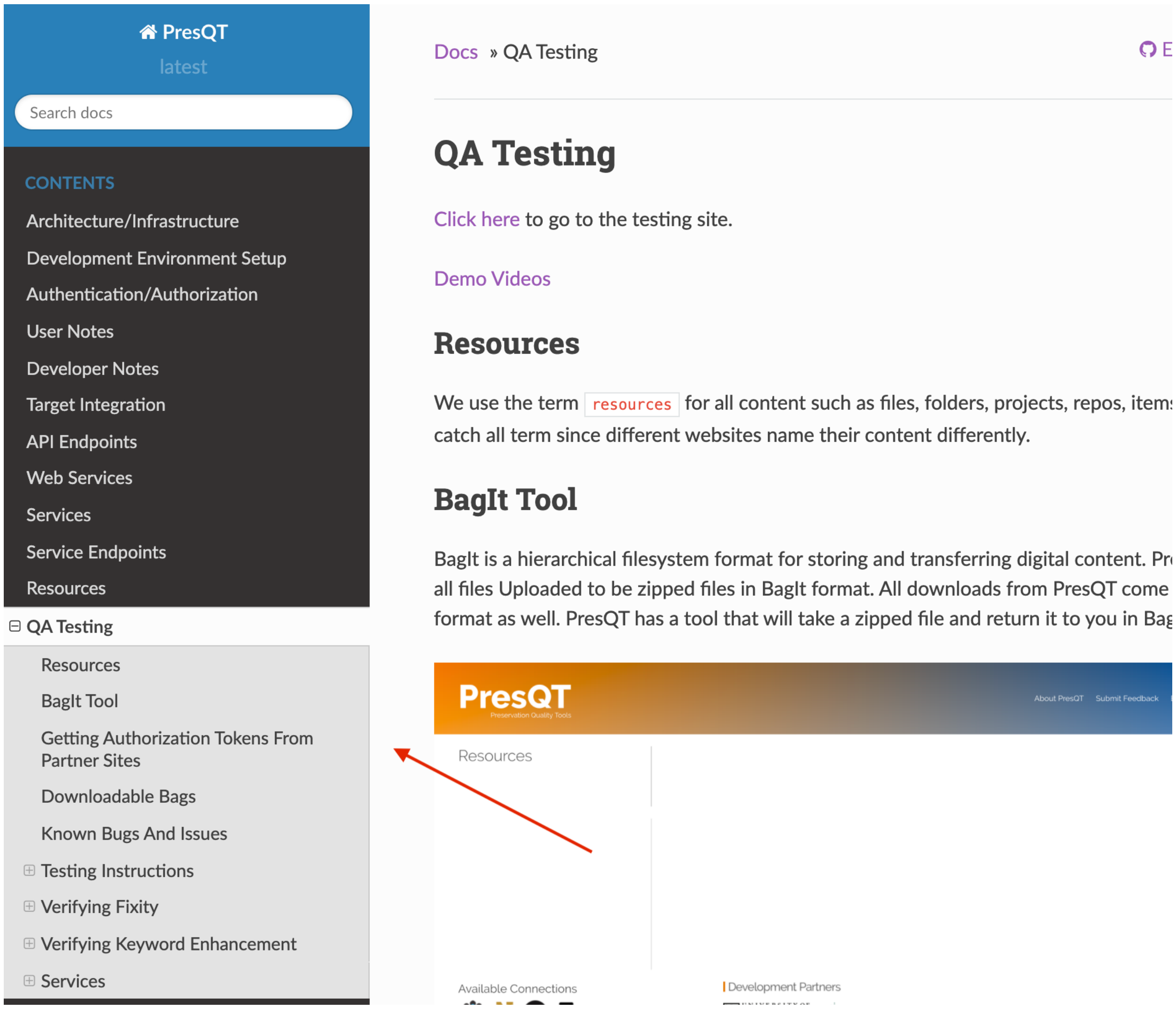
1. Make a POST to `https://presqt-prod.crc.nd.edu/api_v1/targets/osf/resources/` with BagIt #2 to see a new top level container created.

2. Get the id of the new container and make a POST to `https://presqt-prod.crc.nd.edu/api_v1/targets/osf/resources/{resource_id}/` with BagIt #3 and with the 'presqt-file-duplicate-action' set to 'ignore' to see that the duplicate file is found and it's contents are different but the file is updated.
3. Make the same request as 2 but set the header 'presqt-file-duplicate-action' to 'update' to see the file updated.
4. With the same container id make a POST request to `https://presqt-prod.crc.nd.edu/api_v1/targets/osf/resources/{resource_id}/` with BagIt #4 to see new files and folders added to the top level container.
5. A POST request with BagIts 5-7 should return an error with nothing being uploaded.

# CHAPTER TWELVE

# QA TESTING

Be sure to use the table of contents shown here to help navigate the instructions



QA Testing

Click here to go to the testing site.

Demo Videos

Resources

We use the term `resources` for all content such as files, folders, projects, repos, items
catch all term since different websites name their content differently.

BagIt Tool

BagIt is a hierarchical filesystem format for storing and transferring digital content. Pr
all files Uploaded to be zipped files in BagIt format. All downloads from PresQT come
format as well. PresQT has a tool that will take a zipped file and return it to you in Bag

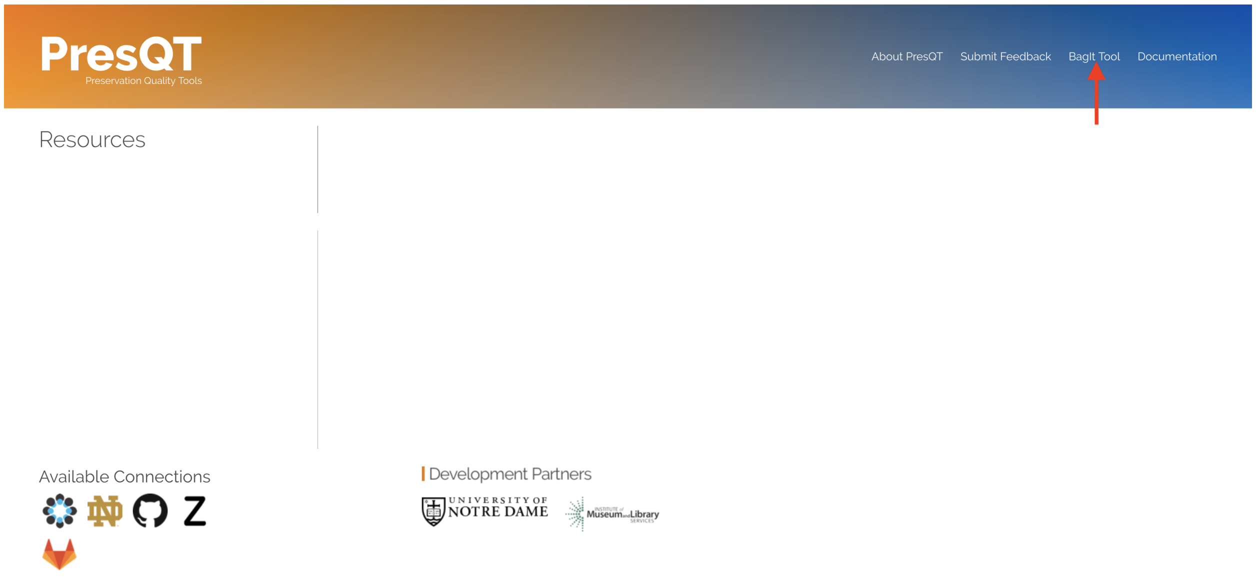
to go to the testing site.

## 12.1 Resources

We use the term `resources` for all content such as files, folders, projects, repos, items, etc. It's a catch all term since different websites name their content differently.

## 12.2 BagIt Tool

BagIt is a hierarchical filesystem format for storing and transferring digital content. PresQT expects all files Uploaded to be zipped files in BagIt format. All downloads from PresQT come in BagIt format as well. PresQT has a tool that will take a zipped file and return it to you in BagIt format.



## 12.3 Getting Authorization Tokens From Partner Sites

An `Authorization Token` is a unique identifier for a user requesting access to a service.

You can for instructions on how to get authorization tokens for each target.

## 12.4 Test Files

Here are some pre-made ZIP files that are in BagIt format that can be downloaded for use with PresQT.

`presqt_Images.zip`

`presqt_MediaFiles.zip`

`presqt_TextFiles.zip`

`presqt_MixedFileTypes.zip`

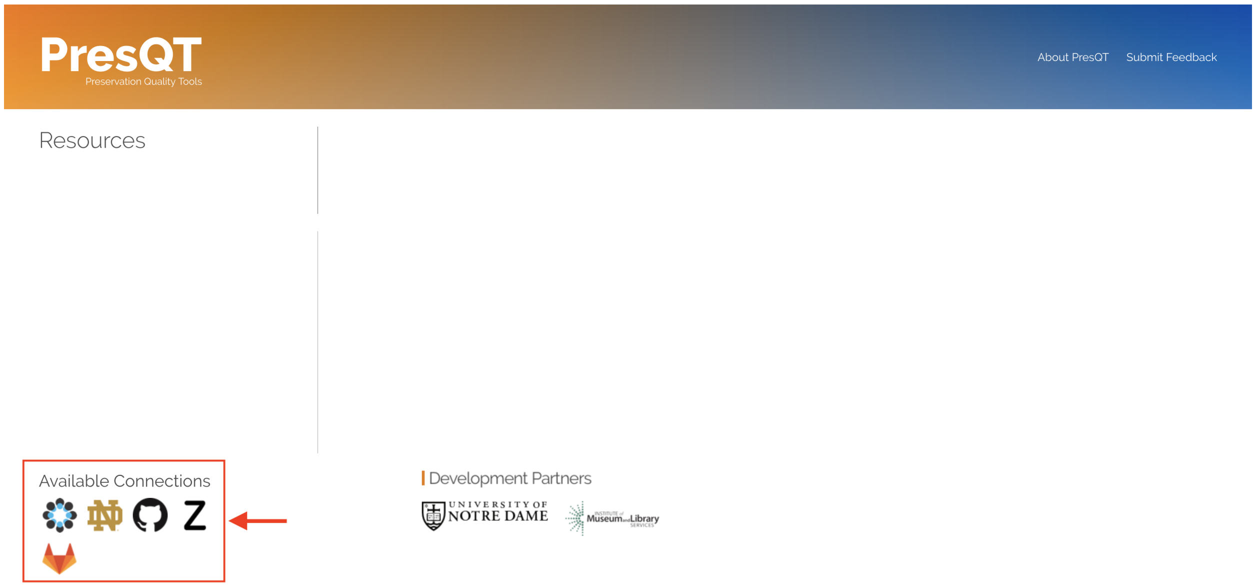
## 12.5 Known Bugs And Issues

- None as of this writing

## 12.6 Testing Instructions

### 12.6.1 Login To Targets From PresQT Demo UI

1. Click on any Target icon under 'Available Connections' to pop open a login window.



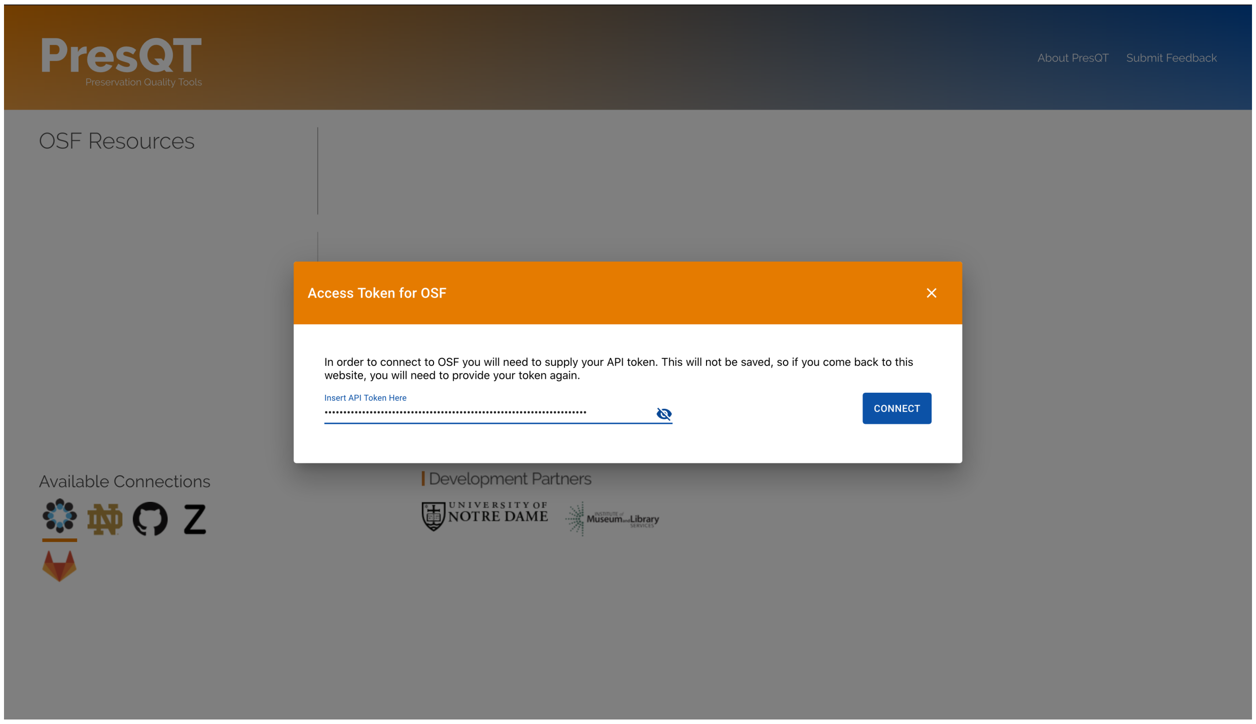
2. Copy your `Authorization Token` for the target and press `Connect`
3. Resources associated with this token will appear on the left side.
4. You can log out of the target and use a different token by pressing the button next to the resources header.

5. To log into a different target simply repeat the process with a different target icon. Once logged in you can switch between targets without having to provide your key.

### 12.6.2 Navigate and Searching The Resource Collection

to go to the testing site.

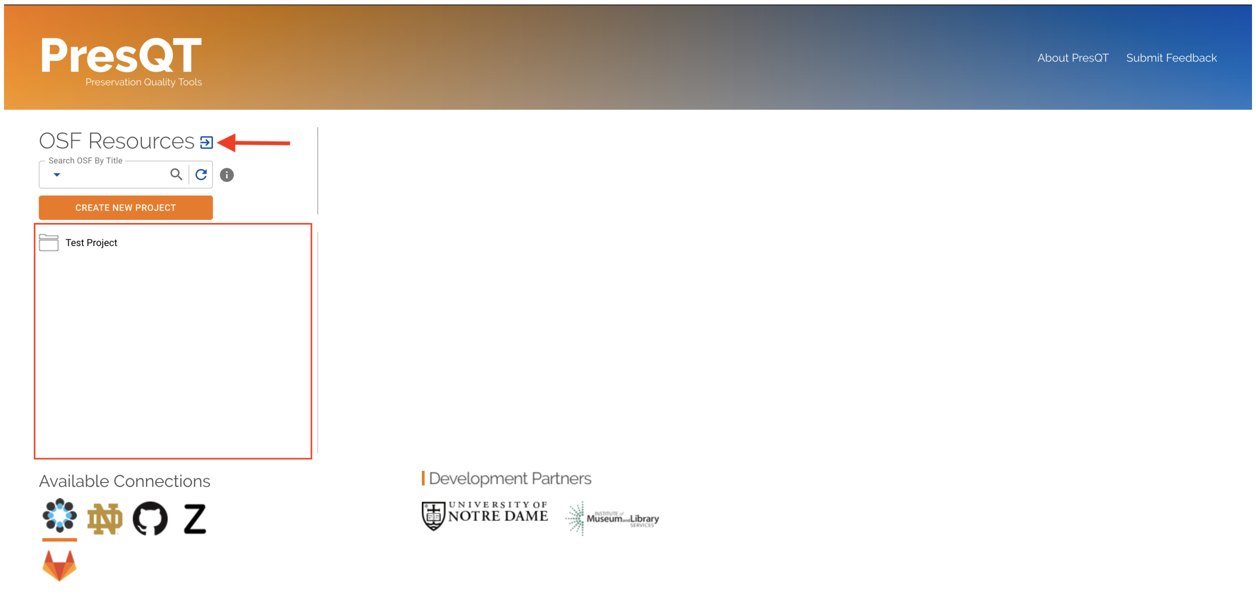
1. After logging in you can navigate through your `Resource Collection` by clicking on the folders and files on the left.
2. Clicking on a resource shows you the `Resource Details` on the right.

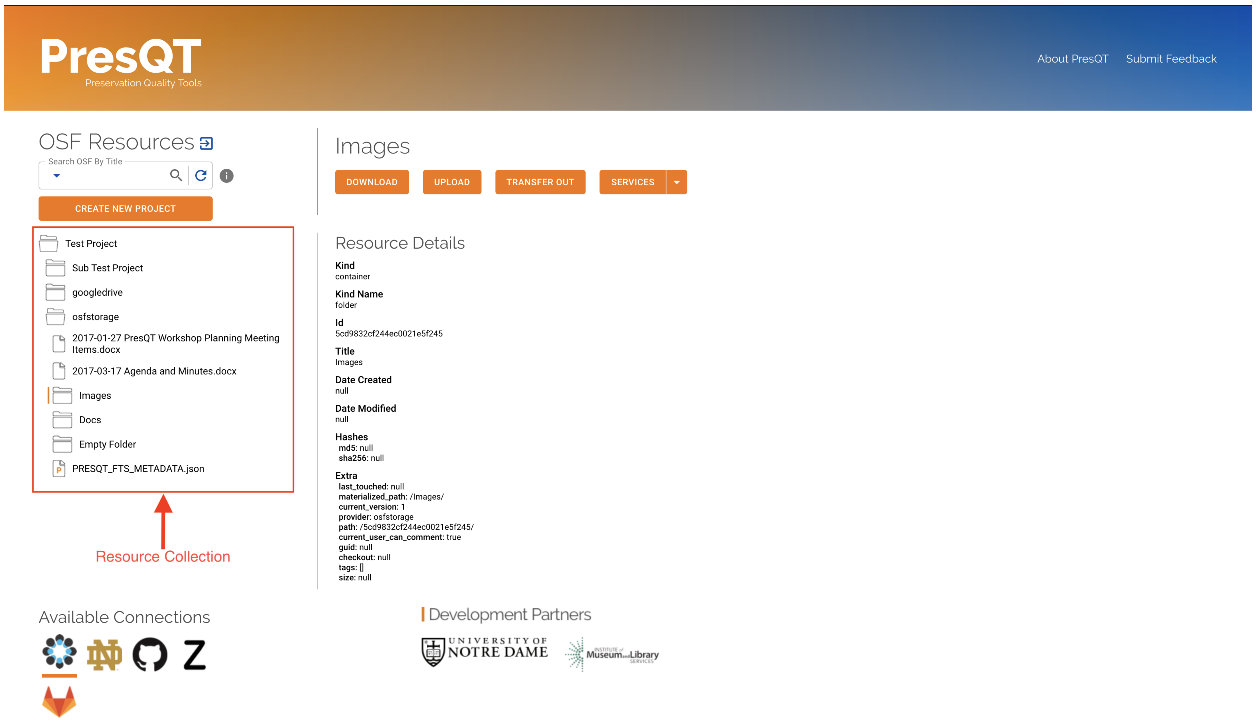


Access Token for OSF

In order to connect to OSF you will need to supply your API token. This will not be saved, so if you come back to this website, you will need to provide your token again.

Insert API Token Here





Resource Details

**Kind**
container

**Kind Name**
folder

**Id**
5cd9832cf244ec0021e5f245

**Title**
Images

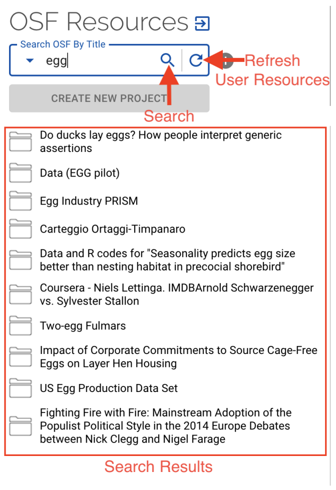
**Date Created**
null

**Date Modified**
null

**Hashes**
**md5:** null
**sha256:** null

**Extra**
**last_touched:** null
**materialized_path:** /Images/
**current_version:** 1
**provider:** osfstorage
**path:** /5cd9832cf244ec0021e5f245/
**current_user_can_comment:** true
**guid:** null
**checkout:** null
**tags:** []
**size:** null

3. Searching for public resources can be accomplished by selecting a search type and then pressing the `search icon`. Public resources will be shown in the `Resource Collection`. 4. You can get back to your resources by pressing the `refresh button`.



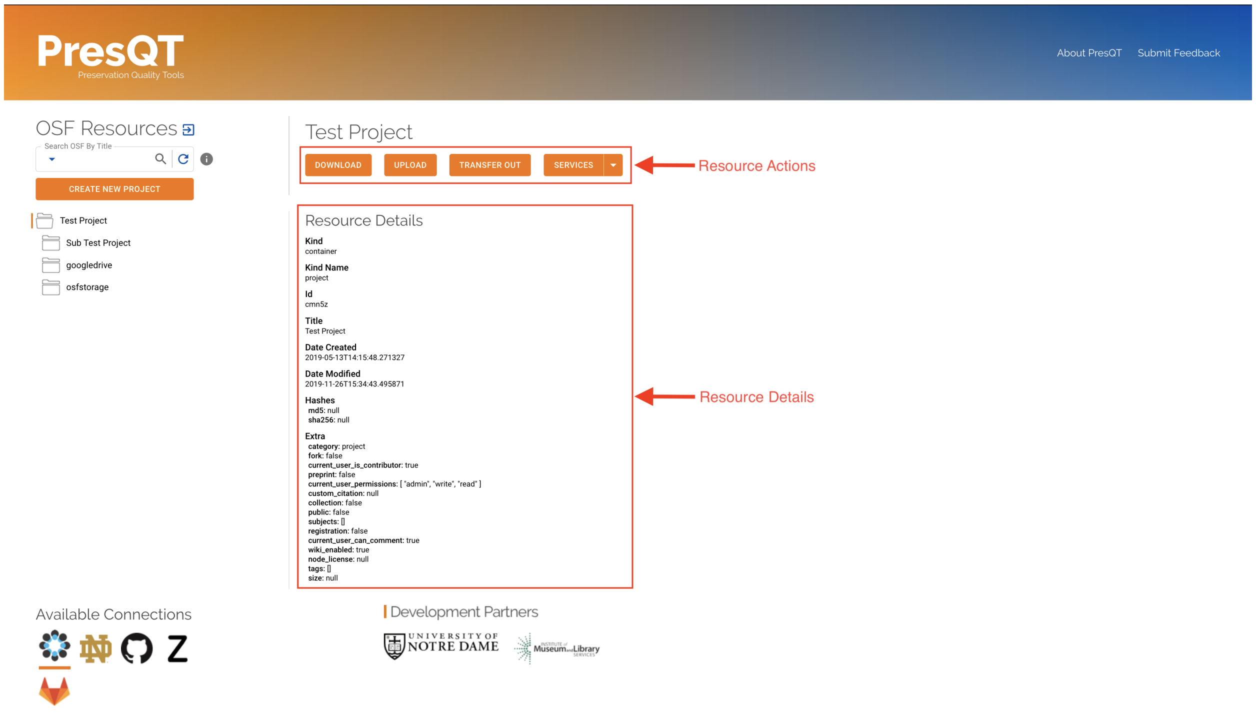
Do ducks lay eggs? How people interpret generic assertions
Data (EGG pilot)
Egg Industry PRISM
Carteggio Ortaggi-Timpanaro
Data and R codes for "Seasonality predicts egg size better than nesting habitat in precocial shorebird"
Coursera - Niels Lettinga. IMDBArnold Schwarzenegger vs. Sylvester Stallon
Two-egg Fulmars
Impact of Corporate Commitments to Source Cage-Free Eggs on Layer Hen Housing
US Egg Production Data Set
Fighting Fire with Fire: Mainstream Adoption of the Populist Political Style in the 2014 Europe Debates between Nick Clegg and Nigel Farage

### 12.6.3 Resource Details And Actions

1. Once you click on a resource you will get its details and buttons for each action available for this resource. If the button is disabled then that action isn't available for that resource.

### 12.6.4 Resource Download

to go to the testing site.



Resource Details

**Kind**
container

**Kind Name**
project

**Id**
cmn5z

**Title**
Test Project

**Date Created**
2019-05-13T14:15:48.271327

**Date Modified**
2019-11-26T15:34:43.495871

**Hashes**
**md5**: null
**sha256**: null

**Extra**
**category**: project
**fork**: false
**current_user_is_contributor**: true
**preprint**: false
**current_user_permissions**: [ "admin", "write", "read" ]
**custom_citation**: null
**collection**: false
**public**: false
**subjects**: []
**registration**: false
**current_user_can_comment**: true
**wiki_enabled**: true
**node_license**: null
**tags**: []
**size**: null

1. To download a resource, first select the resource in the `resource collection` and then click the `Download` action button in the details section.

2. A modal will pop open providing you with transaction details. Click on the `Download` button to start the download.

3. Once the download is complete, the modal will provide you with details about how the download process went.

4. All downloads come in `BagIt format`. After the download is complete, unzip the file, and you will see BagIt specification files. The data you requested to download will reside in the `data` folder.

### 12.6.5 Resource Upload

to go to the testing site.

### 12.6.6 Upload As A New Project

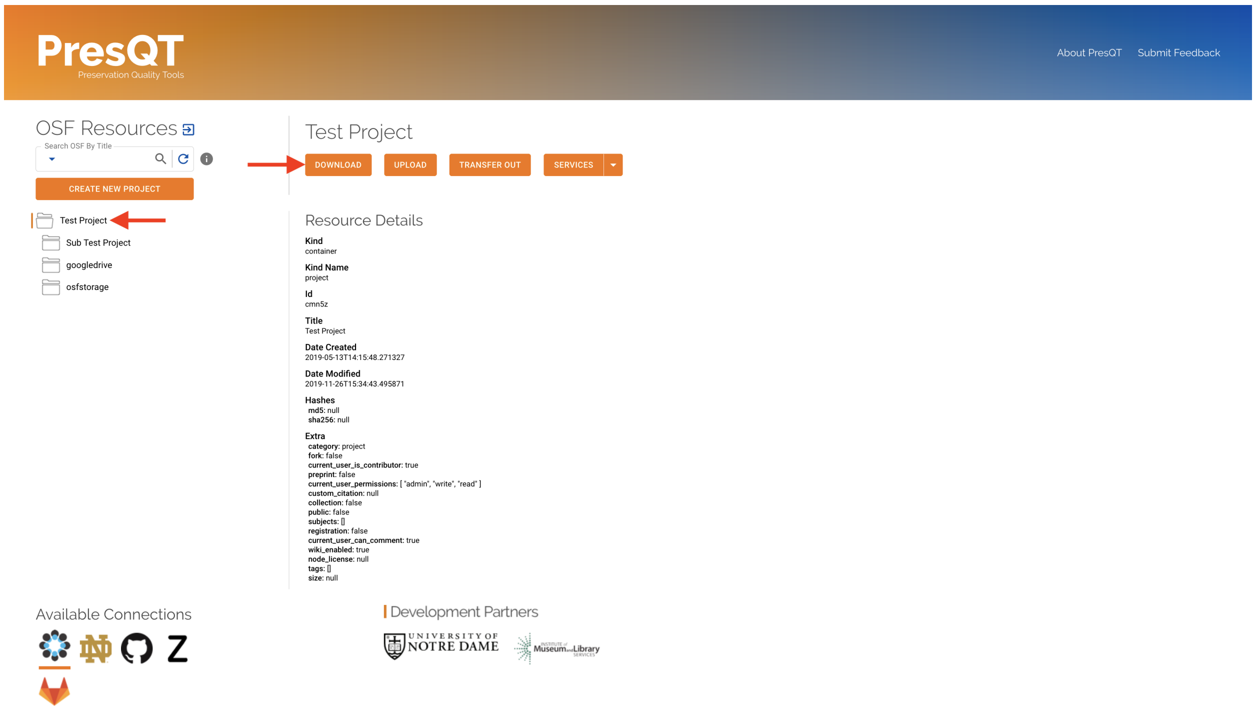
1. To upload to the target as a new project click the `Create New Project` button above the `resource collection`.

2. A modal will pop open with an `upload stepper`. First select the file you'd like to upload. The file must be a zip file who's contents are in valid BagIt format.

3. Next, the modal will display transaction details. Click `Upload File` to begin the upload process.

4. Once the upload is completed, the modal will provide you with details about how the upload process went.

5. You should also see the new uploaded resources appear in the `resource collection`.



Test Project

Resource Details

**Kind**
container

**Kind Name**
project

**Id**
cmn5z

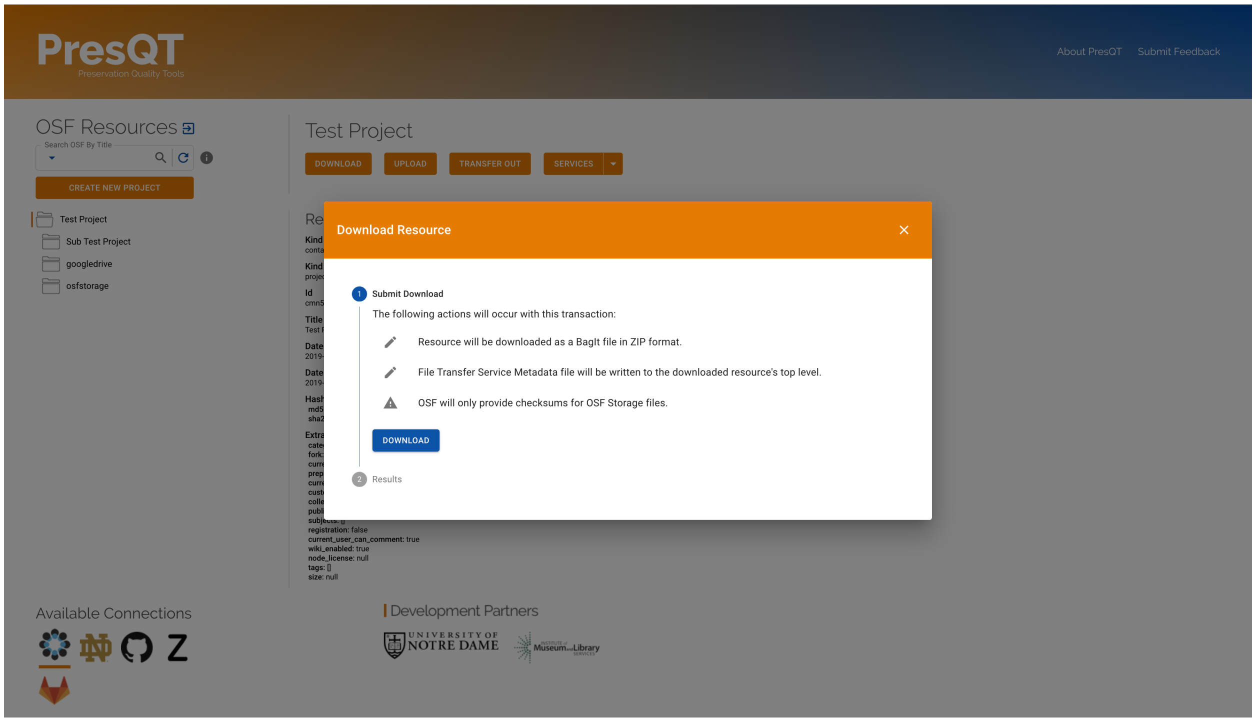
**Title**
Test Project

**Date Created**
2019-05-13T14:15:48.271327

**Date Modified**
2019-11-26T15:34:43.495871

**Hashes**
**md5**: null
**sha256**: null

**Extra**
**category**: project
**fork**: false
**current_user_is_contributor**: true
**preprint**: false
**current_user_permissions**: [ "admin", "write", "read" ]
**custom_citation**: null
**collection**: false
**public**: false
**subjects**: []
**registration**: false
**current_user_can_comment**: true
**wiki_enabled**: true
**node_license**: null
**tags**: []
**size**: null



Download Resource

1 Submit Download

The following actions will occur with this transaction:

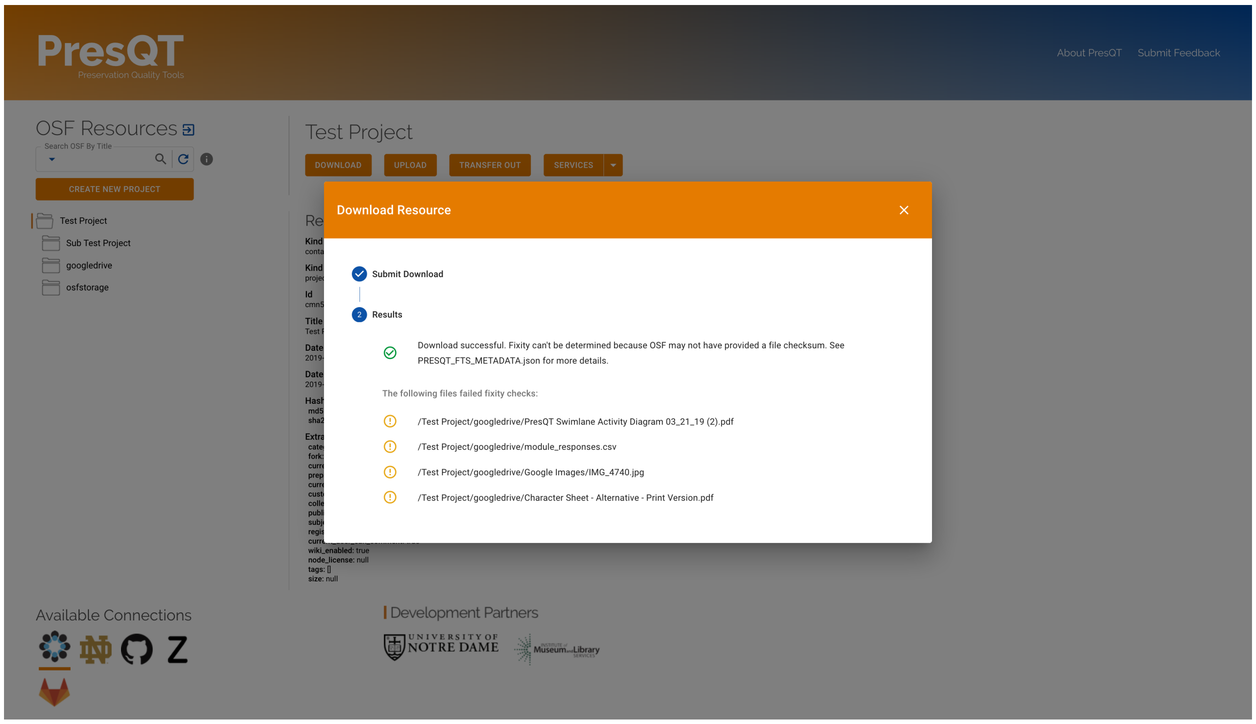
Resource will be downloaded as a BagIt file in ZIP format.

File Transfer Service Metadata file will be written to the downloaded resource's top level.

OSF will only provide checksums for OSF Storage files.

DOWNLOAD

2 Results



Download Resource

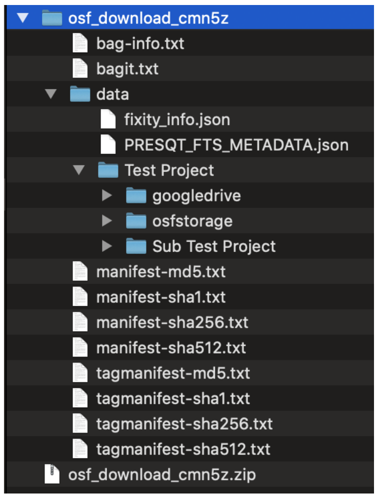
Submit Download

Results

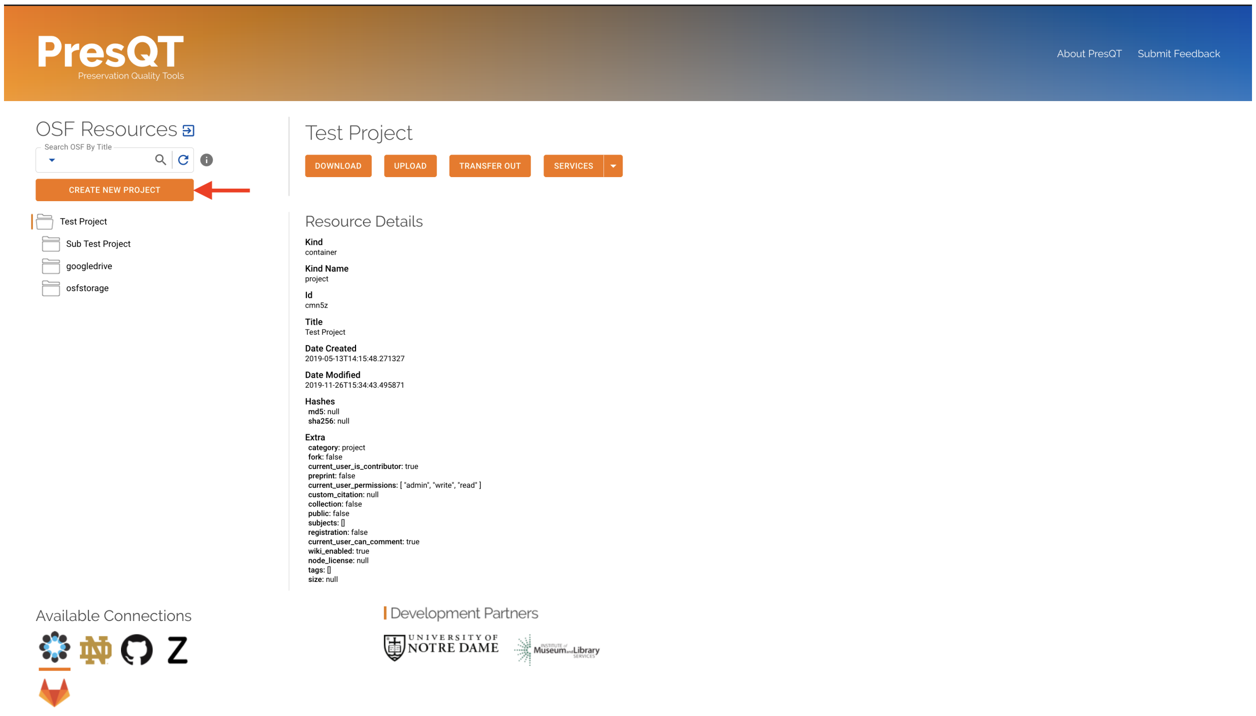
Download successful. Fixity can't be determined because OSF may not have provided a file checksum. See PRESQT_FTS_METADATA.json for more details.

The following files failed fixity checks:

- /Test Project/googledrive/PresQT Swimlane Activity Diagram 03_21_19 (2).pdf
- /Test Project/googledrive/module_responses.csv
- /Test Project/googledrive/Google Images/IMG_4740.jpg
- /Test Project/googledrive/Character Sheet - Alternative - Print Version.pdf



```
▼ osf_download_cmn5z
      bag-info.txt
      bagit.txt
    ▼ data
          fixity_info.json
          PRESQT_FTS_METADATA.json
        ▼ Test Project
            ▶ googledrive
            ▶ osfstorage
            ▶ Sub Test Project
      manifest-md5.txt
      manifest-sha1.txt
      manifest-sha256.txt
      manifest-sha512.txt
      tagmanifest-md5.txt
      tagmanifest-sha1.txt
      tagmanifest-sha256.txt
      tagmanifest-sha512.txt
  osf_download_cmn5z.zip
```



Test Project

Resource Details

**Kind**
container

**Kind Name**
project

**Id**
cmn5z

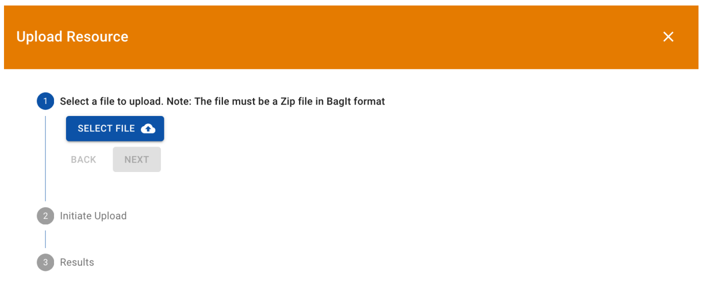
**Title**
Test Project

**Date Created**
2019-05-13T14:15:48.271327

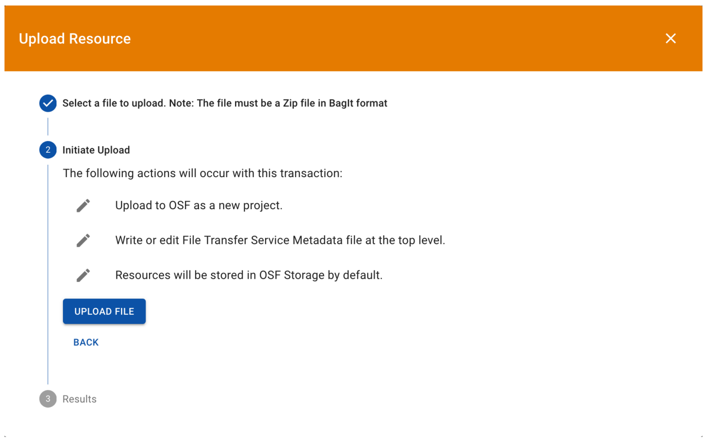
**Date Modified**
2019-11-26T15:34:43.495871

**Hashes**
**md5**: null
**sha256**: null

**Extra**
**category**: project
**fork**: false
**current_user_is_contributor**: true
**preprint**: false
**current_user_permissions**: [ "admin", "write", "read" ]
**custom_citation**: null
**collection**: false
**public**: false
**subjects**: []
**registration**: false
**current_user_can_comment**: true
**wiki_enabled**: true
**node_license**: null
**tags**: []
**size**: null



Select a file to upload. Note: The file must be a Zip file in BagIt format



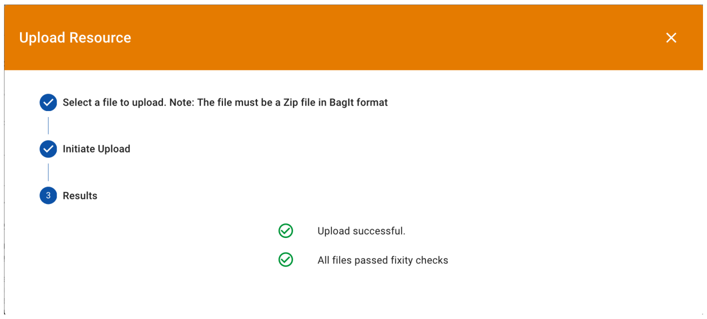
Select a file to upload. Note: The file must be a Zip file in BagIt format

The following actions will occur with this transaction:

Upload to OSF as a new project.

Write or edit File Transfer Service Metadata file at the top level.

Resources will be stored in OSF Storage by default.



Select a file to upload. Note: The file must be a Zip file in BagIt format

Initiate Upload

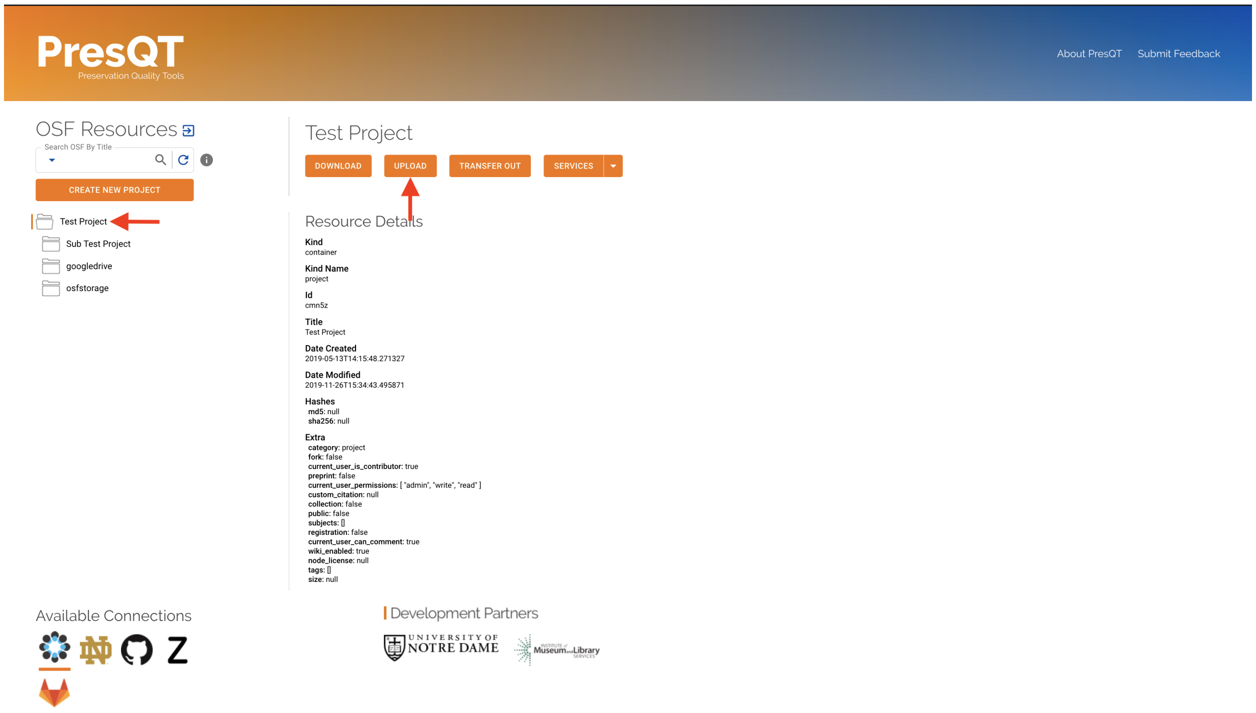
3 Results

Upload successful.

All files passed fixity checks

### 12.6.7 Upload To An Existing Resource

1. To upload a resource, first select the resource in the `resource collection` and then click the `Upload` action button in the details section.



Test Project

Resource Details

**Kind**
container

**Kind Name**
project

**Id**
cmn5z

**Title**
Test Project

**Date Created**
2019-05-13T14:15:48.271327

**Date Modified**
2019-11-26T15:34:43.495871

**Hashes**
**md5**: null
**sha256**: null

**Extra**
**category**: project
**fork**: false
**current_user_is_contributor**: true
**preprint**: false
**current_user_permissions**: [ "admin", "write", "read" ]
**custom_citation**: null
**collection**: false
**public**: false
**subjects**: []
**registration**: false
**current_user_can_comment**: true
**wiki_enabled**: true
**node_license**: null
**tags**: []
**size**: null

2. A modal will pop open with an `upload stepper`. First select the file you'd like to upload. The file must be a zip file who's contents are in valid BagIt format.

3. Select how you want PresQT to handle any duplicate files it finds existing in the resource already. `Ignore` will simply ignore the duplicate. `Update` will update the existing file with the new uploaded file's contents if they differ.

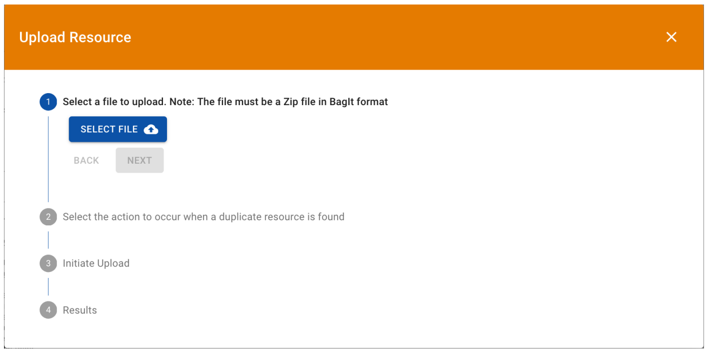
4. Next, the modal will display transaction details. Click `Upload File` to begin the upload process.

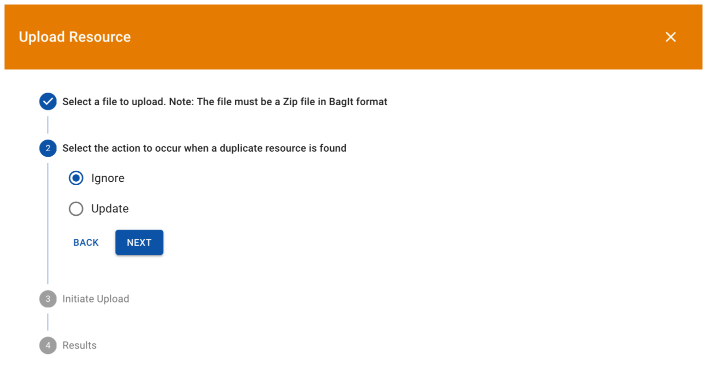
5. Once the upload is completed, the modal will provide you with details about how the upload process went.

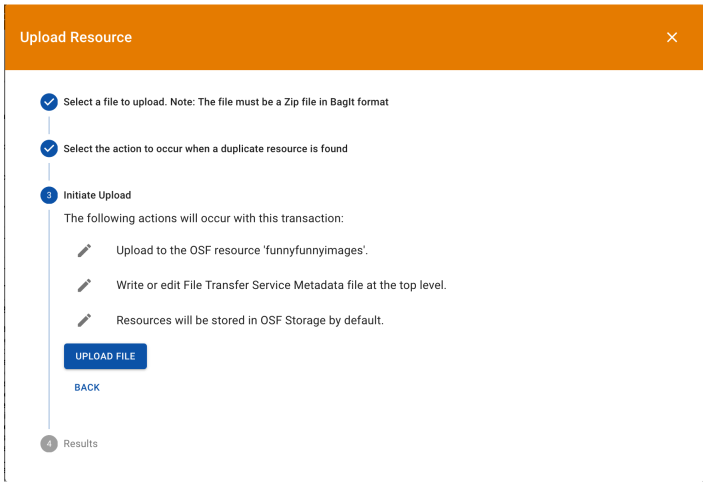
6. You should also see the new uploaded resources appear in the `resource collection`.

### 12.6.8 Resource Transfer

to go to the testing site.


Upload Resource
1 Select a file to upload. Note: The file must be a Zip file in BagIt format
SELECT FILE
BACK
NEXT
2 Select the action to occur when a duplicate resource is found
3 Initiate Upload
4 Results


Upload Resource
Select a file to upload. Note: The file must be a Zip file in BagIt format
2 Select the action to occur when a duplicate resource is found
Ignore
Update
BACK
NEXT
3 Initiate Upload
4 Results

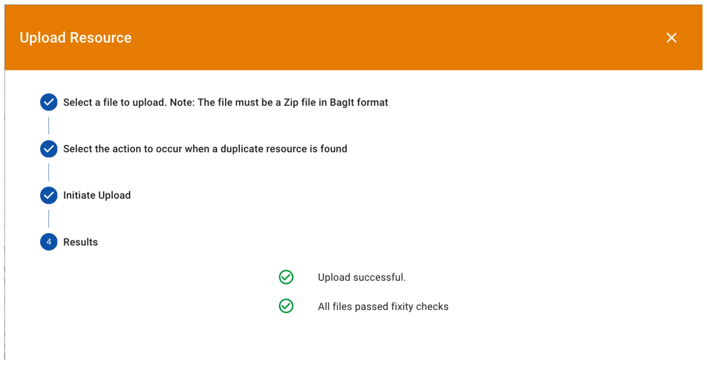

Upload Resource
Select a file to upload. Note: The file must be a Zip file in BagIt format
Select the action to occur when a duplicate resource is found
3 Initiate Upload
UPLOAD FILE
BACK
4 Results

The following actions will occur with this transaction:

Upload to the OSF resource 'funnyfunnyimages'.

Write or edit File Transfer Service Metadata file at the top level.

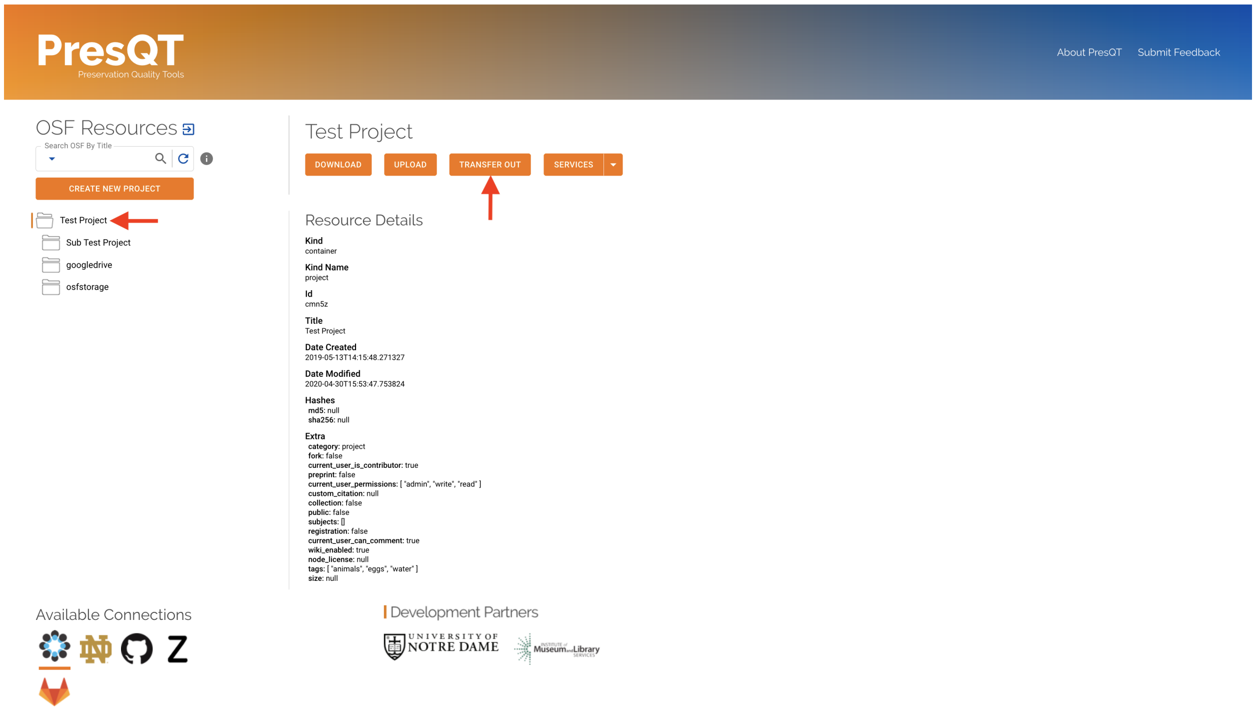
Resources will be stored in OSF Storage by default.


Upload Resource
Select a file to upload. Note: The file must be a Zip file in BagIt format
Select the action to occur when a duplicate resource is found
Initiate Upload
4 Results

Upload successful.

All files passed fixity checks

1. To transfer a resource to another target, first select the resource in the `resource collection` and then click the `Transfer` button in the details section.



Resource Details

Kind
container

Kind Name
project

Id
cmn5z

Title
Test Project

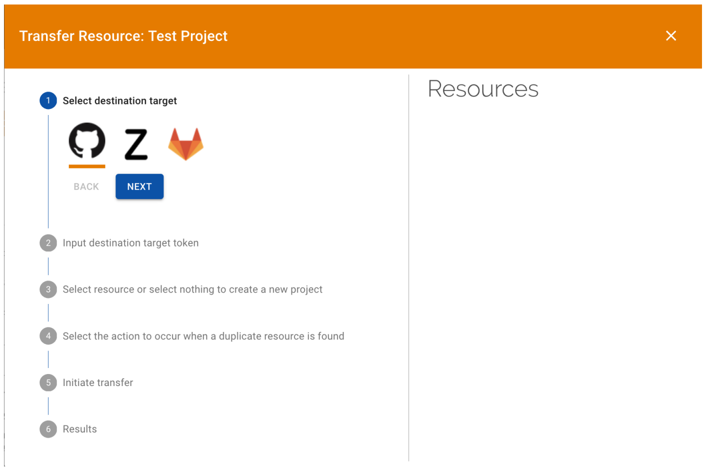
Date Created
2019-05-13T14:15:48.271327

Date Modified
2020-04-30T15:53:47.753824

Hashes
md5: null
sha256: null

Extra
category: project
fork: false
current_user_is_contributor: true
preprint: false
current_user_permissions: [ "admin", "write", "read" ]
custom_citation: null
collection: false
public: false
subjects: []
registration: false
current_user_can_comment: true
wiki_enabled: true
node_license: null
tags: [ "animals", "eggs", "water" ]
size: null

2. A modal will pop open with a `transfer stepper`. First, select the target you want to `transfer to` and press the `Next` button.


Transfer Resource: Test Project
1 Select destination target
BACK
NEXT
2 Input destination target token
3 Select resource or select nothing to create a new project
4 Select the action to occur when a duplicate resource is found
5 Initiate transfer
6 Results
Resources

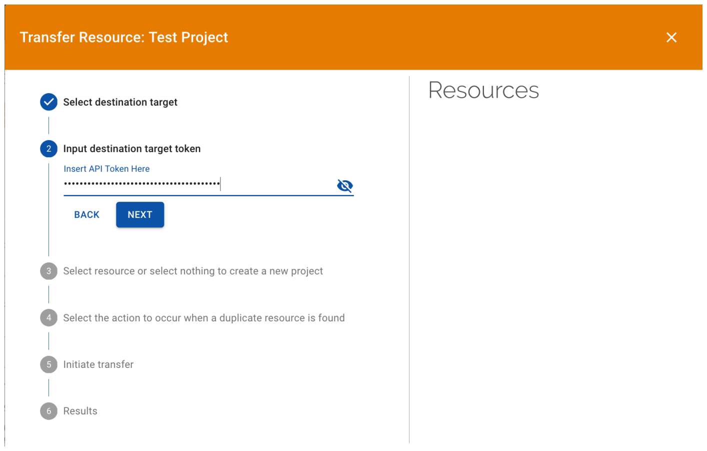
3. Input your token for the target you selected and press the `Next` button.

4. Select the resource you want to transfer to. Don't select any resource if you want to create a new project. Press `Next` once you have made your selection.

5. Select how you want PresQT to handle any duplicate files it finds existing in the resource already. `Ignore` will simply ignore the duplicate. `Update` will update the existing file with the new transferred file's contents if they differ. Press the `Next` button once you've made your selection. If you are making a new project then just press `Next`.

6. Next, the modal will display transaction details. Click `Transfer File` to begin the transfer process.

7. Once the transfer is completed, the modal will provide you with details about how the transfer process went.



Transfer Resource: Test Project

Select destination target

Input destination target token

Insert API Token Here

BACK NEXT

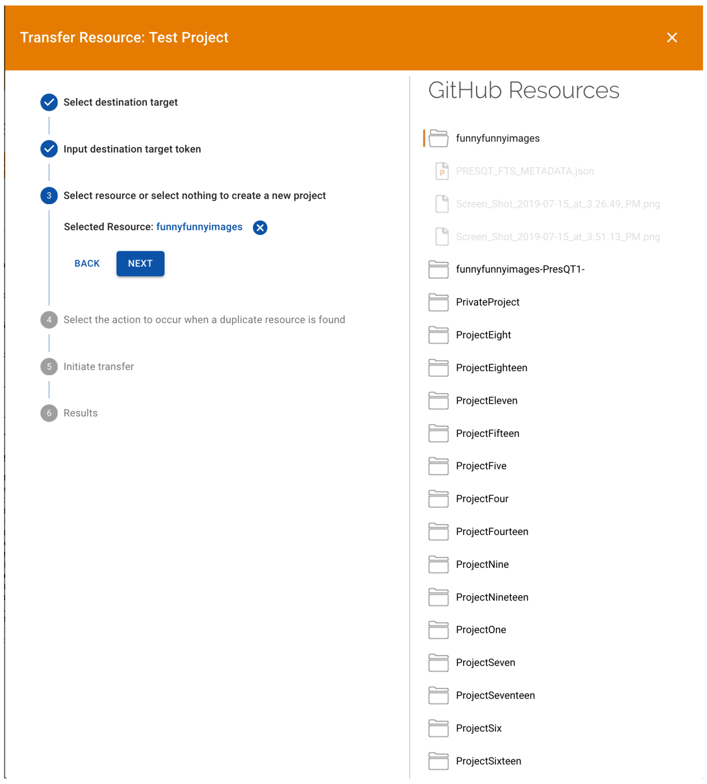
Select resource or select nothing to create a new project

Select the action to occur when a duplicate resource is found

Initiate transfer

Results

Resources



Transfer Resource: Test Project

Select destination target

Input destination target token

Select resource or select nothing to create a new project

Selected Resource: funnyfunnyimages

BACK NEXT

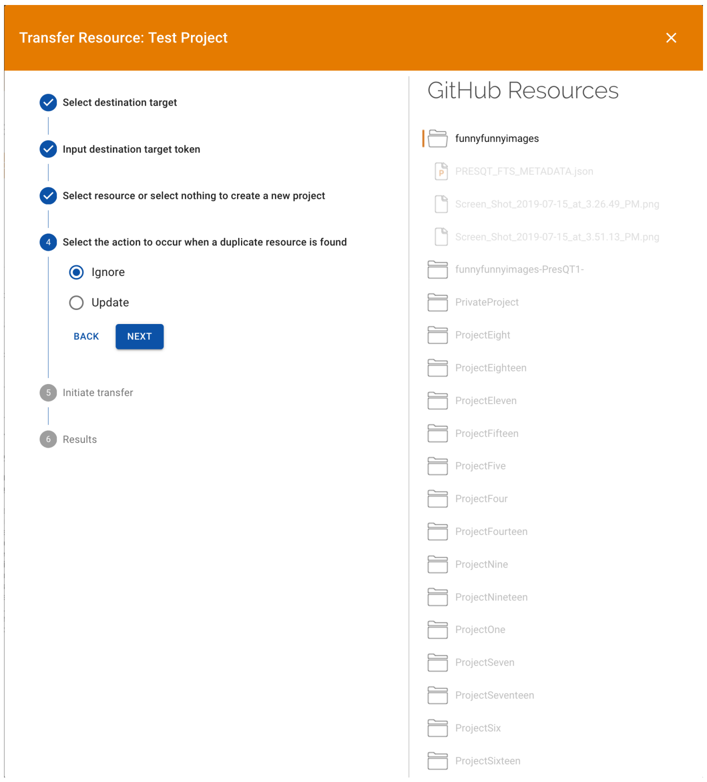
Select the action to occur when a duplicate resource is found

Initiate transfer

Results

GitHub Resources

funnyfunnyimages
PRESQT_FTS_METADATA.json
Screen_Shot_2019-07-15_at_3.26.49_PM.png
Screen_Shot_2019-07-15_at_3.51.13_PM.png
funnyfunnyimages-PresQT1-
PrivateProject
ProjectEight
ProjectEighteen
ProjectEleven
ProjectFifteen
ProjectFive
ProjectFour
ProjectFourteen
ProjectNine
ProjectNineteen
ProjectOne
ProjectSeven
ProjectSeventeen
ProjectSix
ProjectSixteen

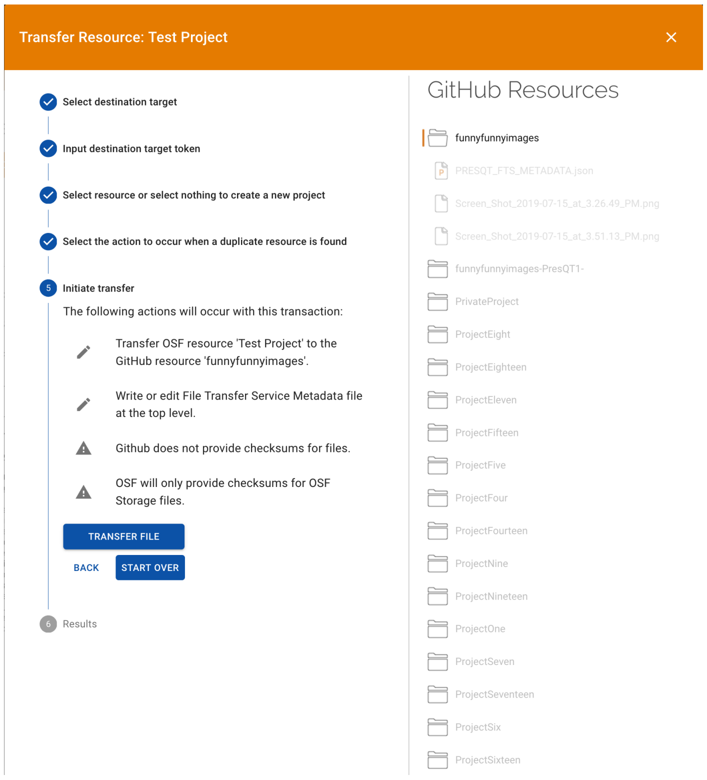


Transfer Resource: Test Project

- Select destination target
- Input destination target token
- Select resource or select nothing to create a new project
- 4 Select the action to occur when a duplicate resource is found
  - [✓ Ignore]
  - Update
  - BACK NEXT
- 5 Initiate transfer
- 6 Results

GitHub Resources

- funnyfunnyimages
  - PRESQT_FTS_METADATA.json
  - Screen_Shot_2019-07-15_at_3.26.49_PM.png
  - Screen_Shot_2019-07-15_at_3.51.13_PM.png
- funnyfunnyimages-PresQT1-
- PrivateProject
- ProjectEight
- ProjectEighteen
- ProjectEleven
- ProjectFifteen
- ProjectFive
- ProjectFour
- ProjectFourteen
- ProjectNine
- ProjectNineteen
- ProjectOne
- ProjectSeven
- ProjectSeventeen
- ProjectSix
- ProjectSixteen



Transfer Resource: Test Project

- Select destination target
- Input destination target token
- Select resource or select nothing to create a new project
- Select the action to occur when a duplicate resource is found
- 5 Initiate transfer

  The following actions will occur with this transaction:

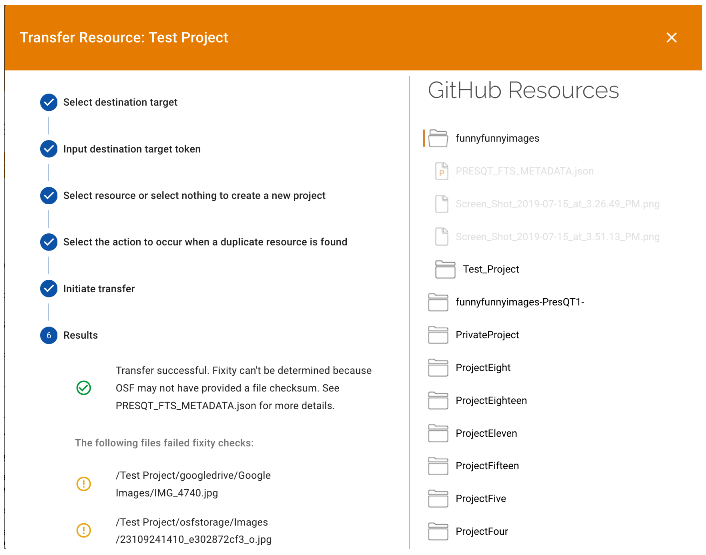
  - Transfer OSF resource 'Test Project' to the GitHub resource 'funnyfunnyimages'.
  - Write or edit File Transfer Service Metadata file at the top level.
  - Github does not provide checksums for files.
  - OSF will only provide checksums for OSF Storage files.

  TRANSFER FILE

  BACK START OVER
- 6 Results

GitHub Resources

- funnyfunnyimages
  - PRESQT_FTS_METADATA.json
  - Screen_Shot_2019-07-15_at_3.26.49_PM.png
  - Screen_Shot_2019-07-15_at_3.51.13_PM.png
- funnyfunnyimages-PresQT1-
- PrivateProject
- ProjectEight
- ProjectEighteen
- ProjectEleven
- ProjectFifteen
- ProjectFive
- ProjectFour
- ProjectFourteen
- ProjectNine
- ProjectNineteen
- ProjectOne
- ProjectSeven
- ProjectSeventeen
- ProjectSix
- ProjectSixteen



Transfer Resource: Test Project

Select destination target

Input destination target token

Select resource or select nothing to create a new project

Select the action to occur when a duplicate resource is found

Initiate transfer

6 Results

Transfer successful. Fixity can't be determined because OSF may not have provided a file checksum. See PRESQT_FTS_METADATA.json for more details.

The following files failed fixity checks:

/Test Project/googledrive/Google Images/IMG_4740.jpg

/Test Project/osfstorage/Images /23109241410_e302872cf3_o.jpg

GitHub Resources

- funnyfunnyimages
  - PRESQT_FTS_METADATA.json
  - Screen_Shot_2019-07-15_at_3.26.49_PM.png
  - Screen_Shot_2019-07-15_at_3.51.13_PM.png
  - Test_Project
- funnyfunnyimages-PresQT1-
- PrivateProject
- ProjectEight
- ProjectEighteen
- ProjectEleven
- ProjectFifteen
- ProjectFive
- ProjectFour

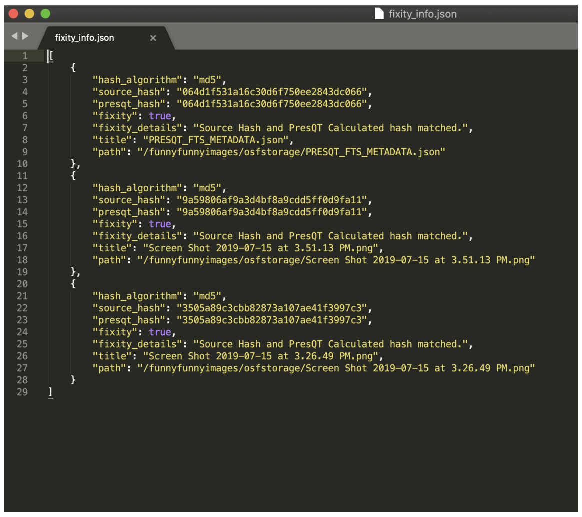
8. You should also see the new transferred resources appear in the modal's `resource collection` on the right.

## 12.7 Verifying Fixity

`Fixity` means the assurance that a digital file has remained unchanged. We determine file fixity at every step along PresQT actions. More details about how PresQT handles fixity can be found Here.

### 12.7.1 Download

All downloads come with a file with detailed fixity information named `fixity_info.json`. This file has an entry for every file involved in the download including each file's checksum hash at the Source Target and the hash calculated on the PresQT servers before sent to the browser for download. To verify fixity remains, the user must calculate the files' hashes on their local machine and compare it to the hashes provided.

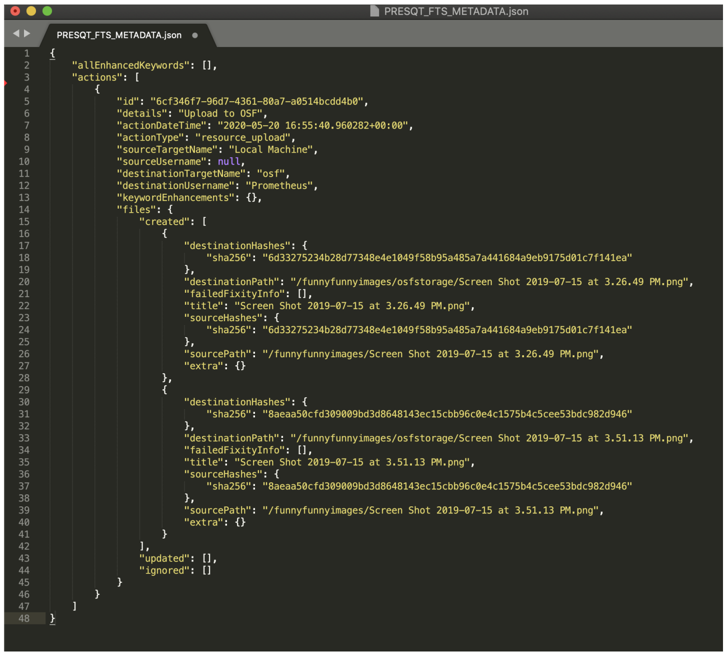


fixity_info.json

```
[
    {
        "hash_algorithm": "md5",
        "source_hash": "064d1f531a16c30d6f750ee2843dc066",
        "presqt_hash": "064d1f531a16c30d6f750ee2843dc066",
        "fixity": true,
        "fixity_details": "Source Hash and PresQT Calculated hash matched.",
        "title": "PRESQT_FTS_METADATA.json",
        "path": "/funnyfunnyimages/osfstorage/PRESQT_FTS_METADATA.json"
    },
    {
        "hash_algorithm": "md5",
        "source_hash": "9a59806af9a3d4bf8a9cdd5ff0d9fa11",
        "presqt_hash": "9a59806af9a3d4bf8a9cdd5ff0d9fa11",
        "fixity": true,
        "fixity_details": "Source Hash and PresQT Calculated hash matched.",
        "title": "Screen Shot 2019-07-15 at 3.51.13 PM.png",
        "path": "/funnyfunnyimages/osfstorage/Screen Shot 2019-07-15 at 3.51.13 PM.png"
    },
    {
        "hash_algorithm": "md5",
        "source_hash": "3505a89c3cbb82873a107ae41f3997c3",
        "presqt_hash": "3505a89c3cbb82873a107ae41f3997c3",
        "fixity": true,
        "fixity_details": "Source Hash and PresQT Calculated hash matched.",
        "title": "Screen Shot 2019-07-15 at 3.26.49 PM.png",
        "path": "/funnyfunnyimages/osfstorage/Screen Shot 2019-07-15 at 3.26.49 PM.png"
    }
]
```

### 12.7.2 Upload

Fixity during upload can be determined by inspecting the `PRESQT_FTS_METADATA.json` file included with every upload. The attribute `failedFixityInfo` in this file will contain the details if the file being uploaded has failed fixity.



```
{
    "allEnhancedKeywords": [],
    "actions": [
        {
            "id": "6cf346f7-96d7-4361-80a7-a0514bcdd4b0",
            "details": "Upload to OSF",
            "actionDateTime": "2020-05-20 16:55:40.960282+00:00",
            "actionType": "resource_upload",
            "sourceTargetName": "Local Machine",
            "sourceUsername": null,
            "destinationTargetName": "osf",
            "destinationUsername": "Prometheus",
            "keywordEnhancements": {},
            "files": {
                "created": [
                    {
                        "destinationHashes": {
                            "sha256": "6d33275234b28d77348e4e1049f58b95a485a7a441684a9eb9175d01c7f141ea"
                        },
                        "destinationPath": "/funnyfunnyimages/osfstorage/Screen Shot 2019-07-15 at 3.26.49 PM.png",
                        "failedFixityInfo": [],
                        "title": "Screen Shot 2019-07-15 at 3.26.49 PM.png",
                        "sourceHashes": {
                            "sha256": "6d33275234b28d77348e4e1049f58b95a485a7a441684a9eb9175d01c7f141ea"
                        },
                        "sourcePath": "/funnyfunnyimages/Screen Shot 2019-07-15 at 3.26.49 PM.png",
                        "extra": {}
                    },
                    {
                        "destinationHashes": {
                            "sha256": "8aeaa50cfd309009bd3d8648143ec15cbb96c0e4c1575b4c5cee53bdc982d946"
                        },
                        "destinationPath": "/funnyfunnyimages/osfstorage/Screen Shot 2019-07-15 at 3.51.13 PM.png",
                        "failedFixityInfo": [],
                        "title": "Screen Shot 2019-07-15 at 3.51.13 PM.png",
                        "sourceHashes": {
                            "sha256": "8aeaa50cfd309009bd3d8648143ec15cbb96c0e4c1575b4c5cee53bdc982d946"
                        },
                        "sourcePath": "/funnyfunnyimages/Screen Shot 2019-07-15 at 3.51.13 PM.png",
                        "extra": {}
                    }
                ],
                "updated": [],
                "ignored": []
            }
        }
    ]
}
```

### 12.7.3 Transfer

Fixity during `Transfer` can be determined the same as `Upload` by inspecting the `PRESQT_FTS_METADATA.json` file in the destination target.

## 12.8 Verifying Keyword Enhancement

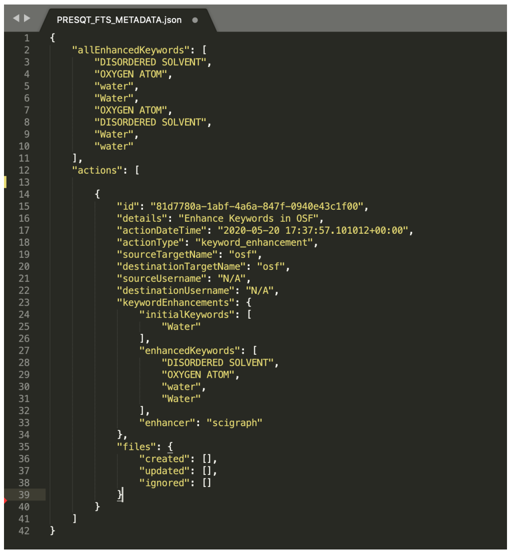
See Here for Keyword Enhancement details.

### 12.8.1 Keyword Enhancement As A Service

Keyword Enhancement as a service will write a new entry to the `PRESQT_FTS_METADATA.json` file in the target. The action entry for keyword enhancement will say exactly which keywords were added during this enhancement.

### 12.8.2 Keyword Enhancement During Transfer

Keyword Enhancement during a transfer will work similarly to `Keyword Enhancement As A Service`. The difference is, for the destination target, the details of keyword enhancement will be located in the transfer action entry instead of there being a new action entry for keyword enhancement.



PRESQT_FTS_METADATA.json

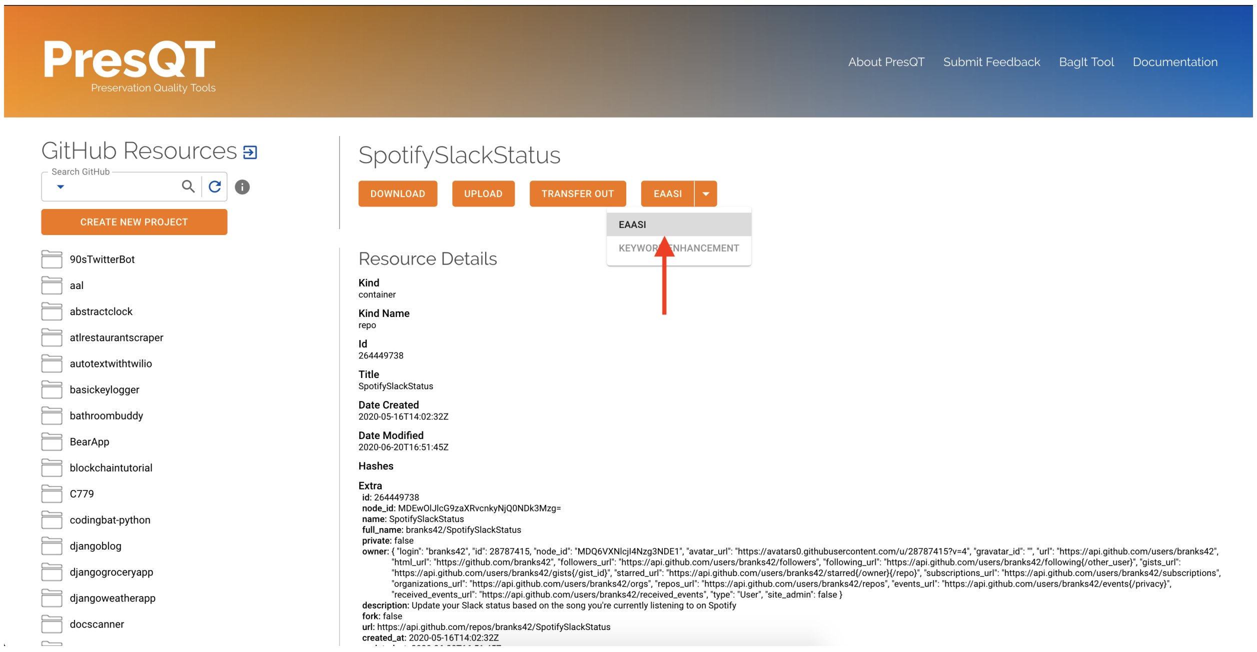
```
{
    "allEnhancedKeywords": [
        "DISORDERED SOLVENT",
        "OXYGEN ATOM",
        "water",
        "Water",
        "OXYGEN ATOM",
        "DISORDERED SOLVENT",
        "Water",
        "water"
    ],
    "actions": [

        {
            "id": "81d7780a-1abf-4a6a-847f-0940e43c1f00",
            "details": "Enhance Keywords in OSF",
            "actionDateTime": "2020-05-20 17:37:57.101012+00:00",
            "actionType": "keyword_enhancement",
            "sourceTargetName": "osf",
            "destinationTargetName": "osf",
            "sourceUsername": "N/A",
            "destinationUsername": "N/A",
            "keywordEnhancements": {
                "initialKeywords": [
                    "Water"
                ],
                "enhancedKeywords": [
                    "DISORDERED SOLVENT",
                    "OXYGEN ATOM",
                    "water",
                    "Water"
                ],
                "enhancer": "scigraph"
            },
            "files": {
                "created": [],
                "updated": [],
                "ignored": []
            }
        }
    ]
}
```

## 12.9 Services

to go to the testing site.

### 12.9.1 Send a Proposal to EaaSI

1. To send a resource to EaaSI, first select the resource in the `resource collection` and then click the `Services` action button in the details section. A drop down menu will appear from where you can select `EaaSI`.



Resource Details

**Kind**
container

**Kind Name**
repo

**Id**
264449738

**Title**
SpotifySlackStatus

**Date Created**
2020-05-16T14:02:32Z

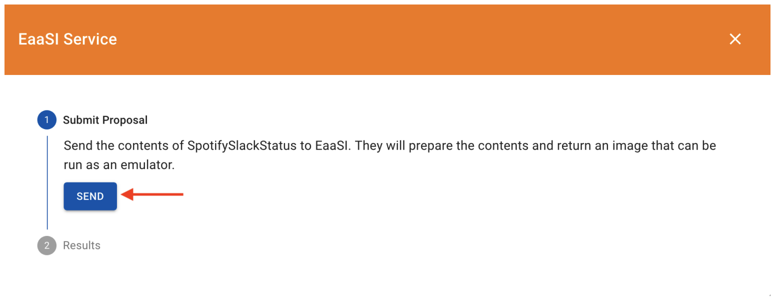
**Date Modified**
2020-06-20T16:51:45Z

**Hashes**

**Extra**
**id:** 264449738
**node_id:** MDEwOlJlcG9zaXRvcnkyNjQ0NDk3Mzg=
**name:** SpotifySlackStatus
**full_name:** branks42/SpotifySlackStatus
**private:** false
**owner:** { "login": "branks42", "id": 28787415, "node_id": "MDQ6VXNlcjI4Nzg3NDE1", "avatar_url": "https://avatars0.githubusercontent.com/u/28787415?v=4", "gravatar_id": "", "url": "https://api.github.com/users/branks42", "html_url": "https://github.com/branks42", "followers_url": "https://api.github.com/users/branks42/followers", "following_url": "https://api.github.com/users/branks42/following{/other_user}", "gists_url": "https://api.github.com/users/branks42/gists{/gist_id}", "starred_url": "https://api.github.com/users/branks42/starred{/owner}{/repo}", "subscriptions_url": "https://api.github.com/users/branks42/subscriptions", "organizations_url": "https://api.github.com/users/branks42/orgs", "repos_url": "https://api.github.com/users/branks42/repos", "events_url": "https://api.github.com/users/branks42/events{/privacy}", "received_events_url": "https://api.github.com/users/branks42/received_events", "type": "User", "site_admin": false }
**description:** Update your Slack status based on the song you're currently listening to on Spotify
**fork:** false
**url:** https://api.github.com/repos/branks42/SpotifySlackStatus
**created_at:** 2020-05-16T14:02:32Z

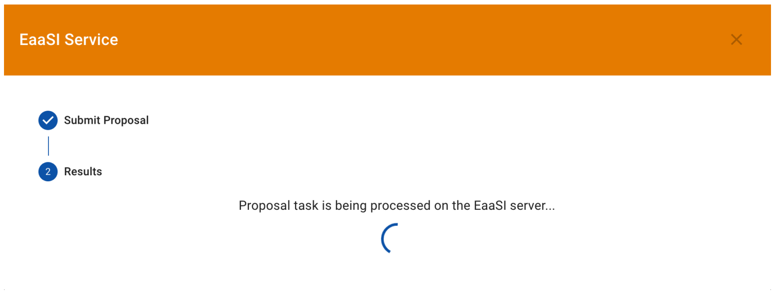
2. A modal will pop open with an `EaaSI stepper`. First read the proposal and ensure the information is correct. Once you have verified that this is what you'd like to do, press the `Send` button.

3. A spinner will keep you informed of where in the process the request is, whether that be on the PresQT server or



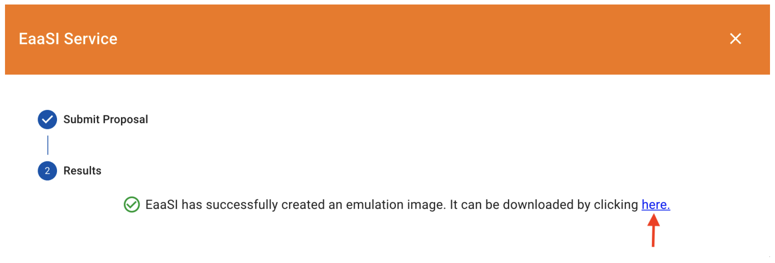
Send the contents of SpotifySlackStatus to EaaSI. They will prepare the contents and return an image that can be run as an emulator.

on EaaSI's.



Proposal task is being processed on the EaaSI server...

4. Once the upload is completed, the modal will provide you with details about how the process went. There will also be a link for you to download the EaaSI created image.



EaaSI has successfully created an emulation image. It can be downloaded by clicking here.

5. You can now open the image and run it however you please. Note: At this point in time, EaaSI's server is only returning cd-rom images for us during testing. The environments will be changed to accurately take into account the files contained within the project as development continues.

### 12.9.2 FAIRshare Evaluator Service

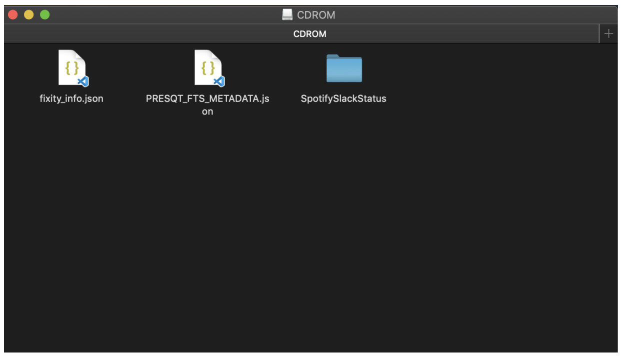
1. To initiate a FAIRshare evaluation, first select the resource in the `resource collection` and then click the `Services` action button in the details section. A drop down menu will appear from where you can select `FAIRshare`.

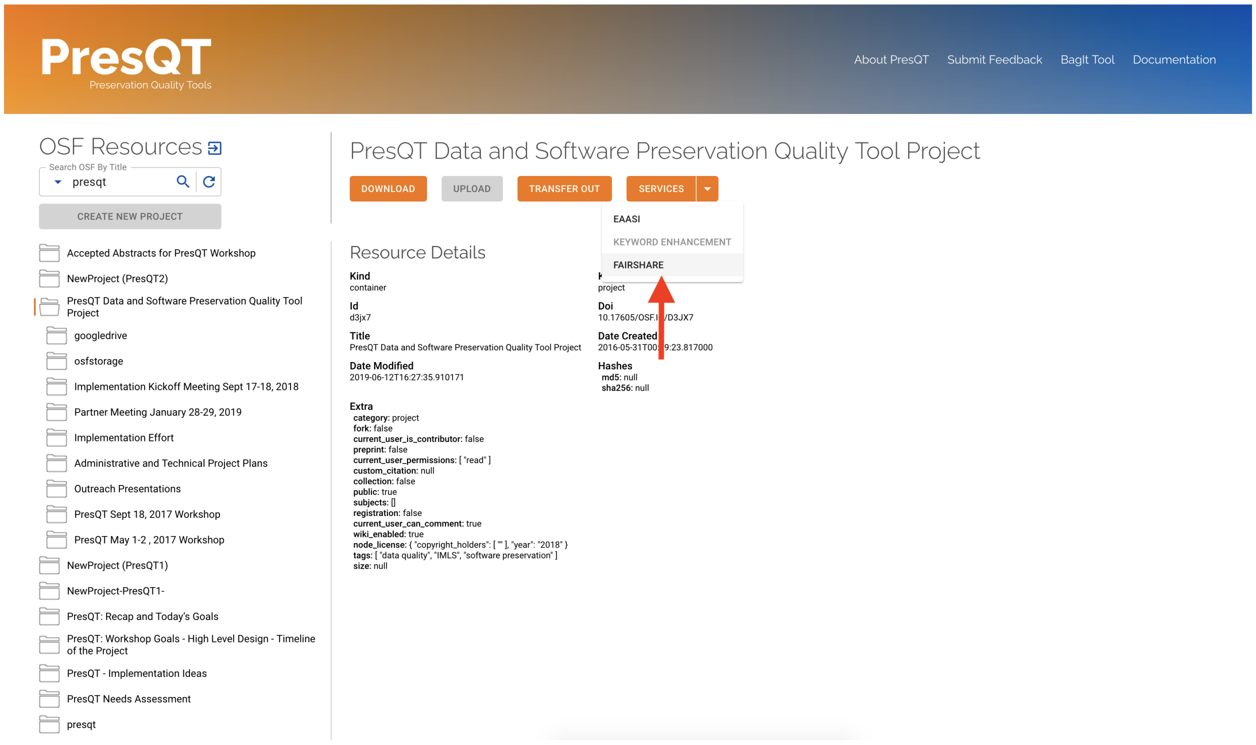
2. A modal will pop open with a `FAIRshare Evaluator Service stepper`. First read the information and ensure the information is correct. Once you have verified that this is what you'd like to do, select the tests you would like to run.

3. Once you have selected the tests you'd like to run, you can choose to opt in for email notifications. When you are ready to run the tests, press the `Evaluate` button.

4. A spinner will let you know that FAIRshare is processing the request. This may take awhile.

5. Once the process is complete, the results will be displayed in a drop down format to be reviewed.





Accepted Abstracts for PresQT Workshop
NewProject (PresQT2)
PresQT Data and Software Preservation Quality Tool Project
googledrive
osfstorage
Implementation Kickoff Meeting Sept 17-18, 2018
Partner Meeting January 28-29, 2019
Implementation Effort
Administrative and Technical Project Plans
Outreach Presentations
PresQT Sept 18, 2017 Workshop
PresQT May 1-2 , 2017 Workshop
NewProject (PresQT1)
NewProject-PresQT1-
PresQT: Recap and Today's Goals
PresQT: Workshop Goals - High Level Design - Timeline of the Project
PresQT - Implementation Ideas
PresQT Needs Assessment
presqt

PresQT Data and Software Preservation Quality Tool Project

Resource Details

**Kind**
container

**Id**
d3jx7

**Title**
PresQT Data and Software Preservation Quality Tool Project

**Date Modified**
2019-06-12T16:27:35.910171

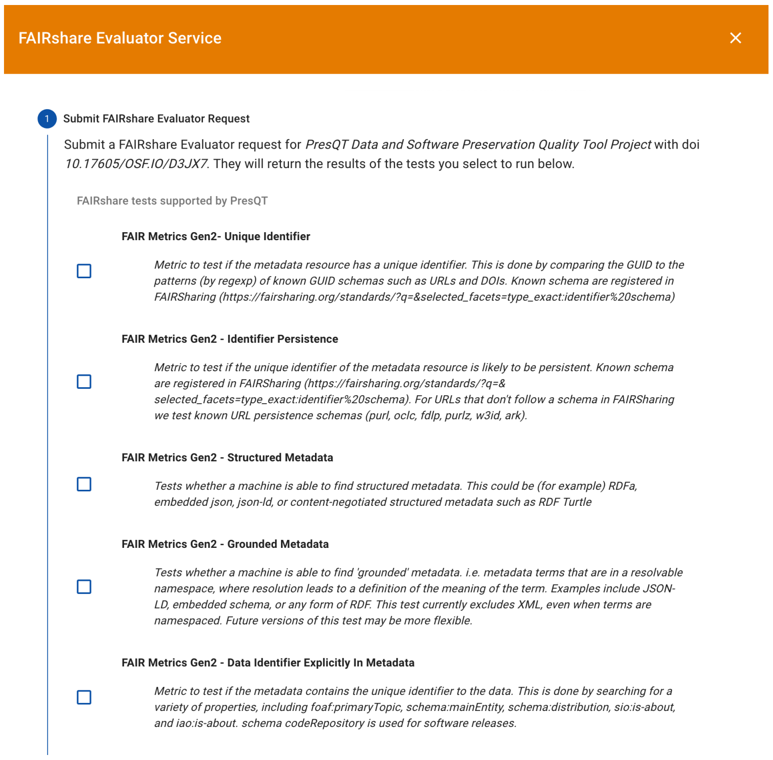
**Extra**
**category:** project
**fork:** false
**current_user_is_contributor:** false
**preprint:** false
**current_user_permissions:** [ "read" ]
**custom_citation:** null
**collection:** false
**public:** true
**subjects:** []
**registration:** false
**current_user_can_comment:** true
**wiki_enabled:** true
**node_license:** { "copyright_holders": [ "" ], "year": "2018" }
**tags:** [ "data quality", "IMLS", "software preservation" ]
**size:** null

project

**Doi**
10.17605/OSF.I /D3JX7

**Date Created**
2016-05-31T00: 9:23.817000

**Hashes**
**md5:** null
**sha256:** null



FAIRshare Evaluator Service

1 Submit FAIRshare Evaluator Request

Submit a FAIRshare Evaluator request for *PresQT Data and Software Preservation Quality Tool Project* with doi *10.17605/OSF.IO/D3JX7*. They will return the results of the tests you select to run below.

FAIRshare tests supported by PresQT

☐ **FAIR Metrics Gen2- Unique Identifier**

*Metric to test if the metadata resource has a unique identifier. This is done by comparing the GUID to the patterns (by regexp) of known GUID schemas such as URLs and DOIs. Known schema are registered in FAIRSharing (https://fairsharing.org/standards/?q=&selected_facets=type_exact:identifier%20schema)*

☐ **FAIR Metrics Gen2 - Identifier Persistence**

*Metric to test if the unique identifier of the metadata resource is likely to be persistent. Known schema are registered in FAIRSharing (https://fairsharing.org/standards/?q=&selected_facets=type_exact:identifier%20schema). For URLs that don't follow a schema in FAIRSharing we test known URL persistence schemas (purl, oclc, fdlp, purlz, w3id, ark).*

☐ **FAIR Metrics Gen2 - Structured Metadata**

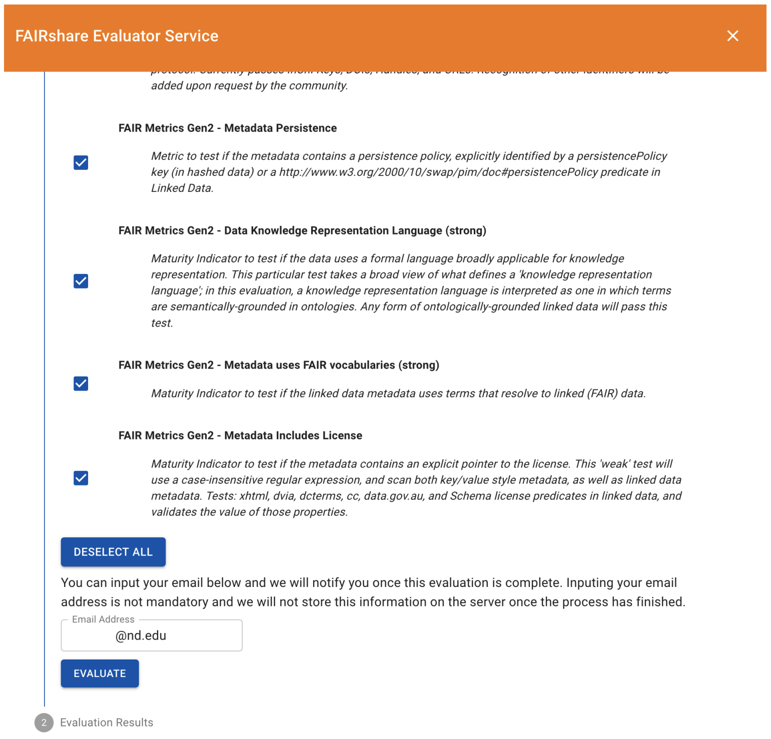
*Tests whether a machine is able to find structured metadata. This could be (for example) RDFa, embedded json, json-ld, or content-negotiated structured metadata such as RDF Turtle*

☐ **FAIR Metrics Gen2 - Grounded Metadata**

*Tests whether a machine is able to find 'grounded' metadata. i.e. metadata terms that are in a resolvable namespace, where resolution leads to a definition of the meaning of the term. Examples include JSON-LD, embedded schema, or any form of RDF. This test currently excludes XML, even when terms are namespaced. Future versions of this test may be more flexible.*

☐ **FAIR Metrics Gen2 - Data Identifier Explicitly In Metadata**

*Metric to test if the metadata contains the unique identifier to the data. This is done by searching for a variety of properties, including foaf:primaryTopic, schema:mainEntity, schema:distribution, sio:is-about, and iao:is-about. schema codeRepository is used for software releases.*



FAIRshare Evaluator Service

*added upon request by the community.*

☑ **FAIR Metrics Gen2 - Metadata Persistence**

*Metric to test if the metadata contains a persistence policy, explicitly identified by a persistencePolicy key (in hashed data) or a http://www.w3.org/2000/10/swap/pim/doc#persistencePolicy predicate in Linked Data.*

☑ **FAIR Metrics Gen2 - Data Knowledge Representation Language (strong)**

*Maturity Indicator to test if the data uses a formal language broadly applicable for knowledge representation. This particular test takes a broad view of what defines a 'knowledge representation language'; in this evaluation, a knowledge representation language is interpreted as one in which terms are semantically-grounded in ontologies. Any form of ontologically-grounded linked data will pass this test.*

☑ **FAIR Metrics Gen2 - Metadata uses FAIR vocabularies (strong)**

*Maturity Indicator to test if the linked data metadata uses terms that resolve to linked (FAIR) data.*

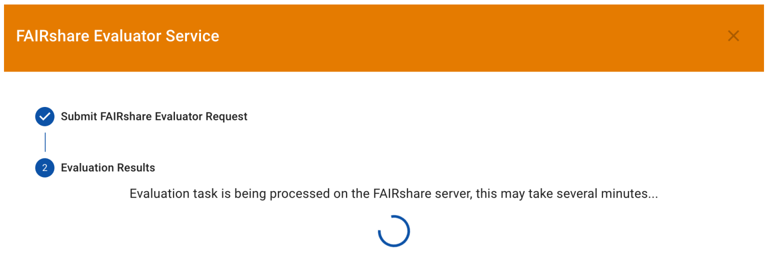
☑ **FAIR Metrics Gen2 - Metadata Includes License**

*Maturity Indicator to test if the metadata contains an explicit pointer to the license. This 'weak' test will use a case-insensitive regular expression, and scan both key/value style metadata, as well as linked data metadata. Tests: xhtml, dvia, dcterms, cc, data.gov.au, and Schema license predicates in linked data, and validates the value of those properties.*

You can input your email below and we will notify you once this evaluation is complete. Inputing your email address is not mandatory and we will not store this information on the server once the process has finished.

Email Address: @nd.edu

2 Evaluation Results

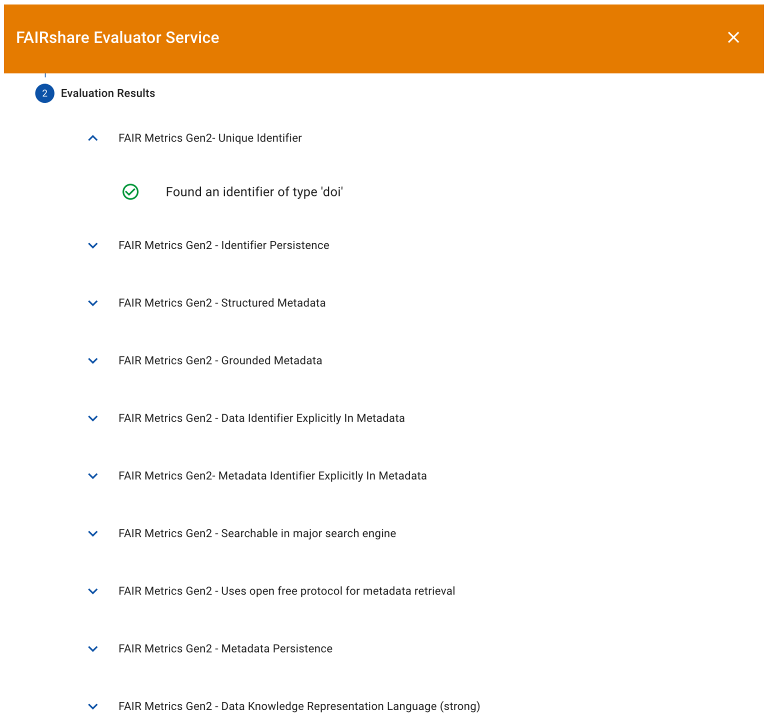


FAIRshare Evaluator Service

Submit FAIRshare Evaluator Request

Evaluation Results

Evaluation task is being processed on the FAIRshare server, this may take several minutes...



FAIRshare Evaluator Service

Evaluation Results

FAIR Metrics Gen2- Unique Identifier

Found an identifier of type 'doi'

FAIR Metrics Gen2 - Identifier Persistence

FAIR Metrics Gen2 - Structured Metadata

FAIR Metrics Gen2 - Grounded Metadata

FAIR Metrics Gen2 - Data Identifier Explicitly In Metadata

FAIR Metrics Gen2- Metadata Identifier Explicitly In Metadata

FAIR Metrics Gen2 - Searchable in major search engine

FAIR Metrics Gen2 - Uses open free protocol for metadata retrieval

FAIR Metrics Gen2 - Metadata Persistence

FAIR Metrics Gen2 - Data Knowledge Representation Language (strong)

# CHAPTER THIRTEEN

# OTHER INTEGRATIONS

## 13.1 Whole Tale Integration Proposal

Whole Tale (http://www.wholetale.org) is a platform for the creation, publication, and re-execution of reproducible computational artifacts. Researchers can create new *tales* that contain the code, data, workflow, and information about the computational environment required to reproduce their analysis. Tales have basic metadata including authors, title, keywords, description, and related identifiers (cited derived from). Tales can be published to archival repositories including DataONE network members and Zenodo.

Whole Tale is an integration partner for the PresQT project. As part of the integration testing process, we explored two different use cases:

- Publish Tales from the WT platform to Zenodo and CurateND using the PresQT APIs
- Import an OSF project into WT using the PresQT APIs

**BagIt Serialization**

Tales are exported as BDBag-compatible bags with the following structure:

```
5e696df5f1c291f11ae9e1a8/
  README.md                <-- Top-level readme (Tag file)
  bagit.txt
  bag-info.txt
  fetch.txt                <-- Fetch file
  manifest-md5.txt
  manifest-sha256.txt
  run-local.sh             <-- Script to run Tale locally (Tag file)
  tagmanifest-md5.txt
  tagmanifest-sha256.txt
  data/
    workspace/             <-- Tale workspace (user code, data, etc)
      environment.yml
      index.ipynb
    LICENSE                <-- Tale license (Tag file)
  metadata/                <-- Tale metadata (Tag directory)
      environment.json
      manifest.json
```

The Tale bag is not currently compatible with PresQT, since PresQT currently requires a single top-level folder in the payload representing the project. The folder name is used as the dataset title when publishing to a target. Since the Tale bag has a separate structure, one option is to double-bag. In the following example, the above Tale is zipped and bagged:

```
presqt_bag/
  bagit.txt
  bag-info.txt
  manifest-md5.txt
  manifest-sha256.txt
  tagmanifest-md5.txt
  tagmanifest-sha256.txt
  data/
    Mapping Estimated Water Usage/
      5e7e10163632f4f0c84c51a8.zip
```

**Publishing to Zenodo**

The following figures illustrate the Tale as published to Zenodo using the WT internal integration and using PresQT.

The first images illustrates a tale published to Zenodo using the WT internal integration. The title, author, and description are all provided by the user during tale creation. The WT platform adds a note with a custom link allowing the user to import and re-execute the tale in the WT system. WT also supports related identifiers, license, and keywork metadata. The tale is published as a zipped bag.

The second image illustrates a tale published to Zenodo using the PresQT integration. The user is directed to the draft dataset creation form where they are required to manually enter relevant metadata.

**Metadata Support**

To support our current Zenodo integration using the PresQT system would require the ability to specify additional metadata during dataset creation. We use the following fields:

- Title
- Creator/Authors (first name, last name, ORCID)
- Publication date/date
- Description
- Subject/keywords
- Rights/license
- Related identifiers (Cites, Derived From)
- References
- Notes

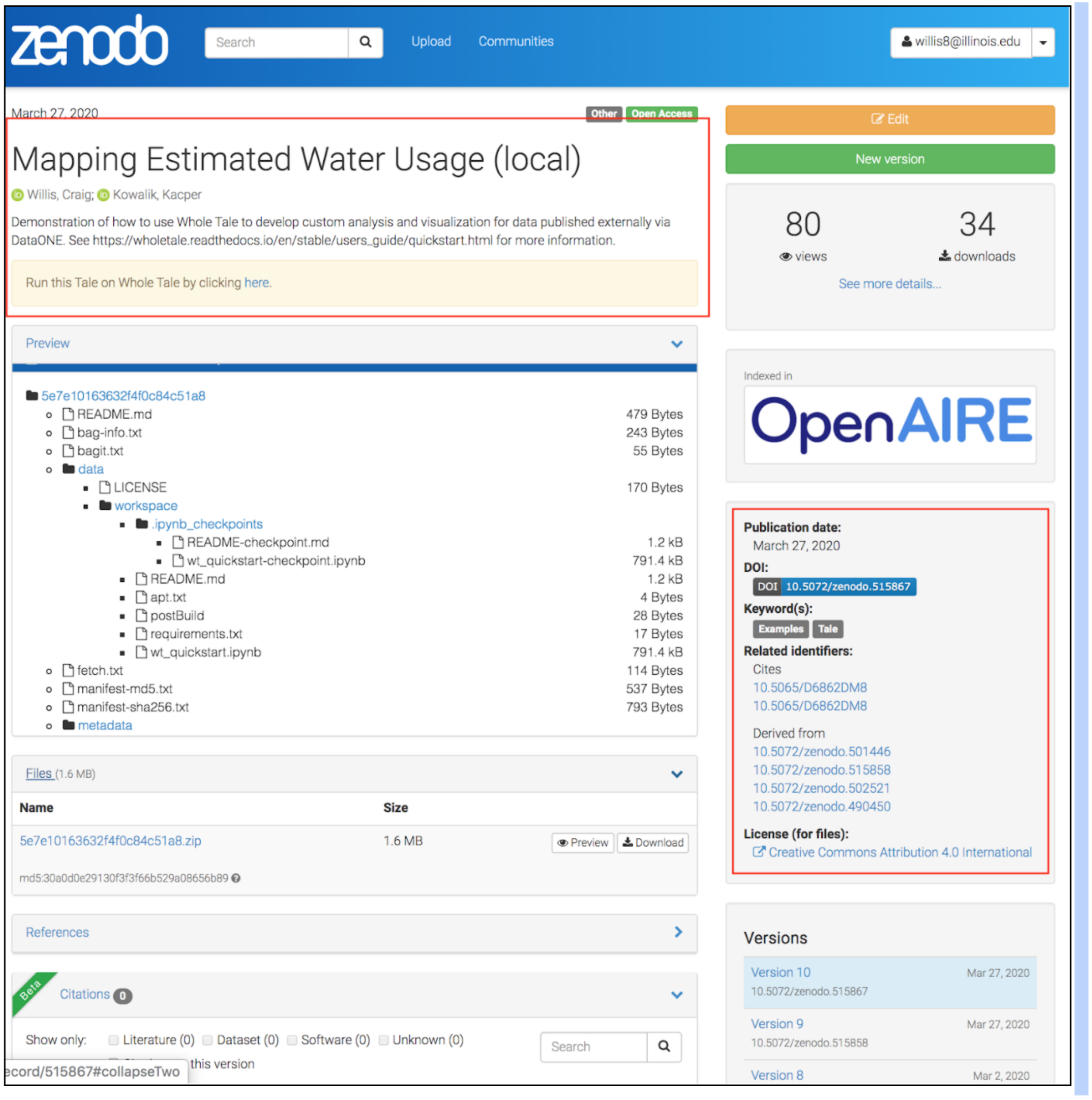
The notes field is important as it provides a way for us to embed a link in the record that allows users to easily re-import and run the published tale.

For more information, see https://github.com/whole-tale/serialization-format/

## 13.2 Whole Tale Integration Implementation

An ‘extra_metadata’ field has been added to the PRESQT_FTS_METADATA.json. To get these extra metadata fields to the new resource being created, the uploaded resources must have a PRESQT_FTS_METADATA.json file at the highest level.

The following is an example:



March 27, 2020

Mapping Estimated Water Usage (local)

Willis, Craig; Kowalik, Kacper

Demonstration of how to use Whole Tale to develop custom analysis and visualization for data published externally via DataONE. See https://wholetale.readthedocs.io/en/stable/users_guide/quickstart.html for more information.

Run this Tale on Whole Tale by clicking here.

80 views 34 downloads

See more details...

```
5e7e10163632f4f0c84c51a8
  ◦ README.md                                   479 Bytes
  ◦ bag-info.txt                                243 Bytes
  ◦ bagit.txt                                    55 Bytes
  ◦ data
      ▪ LICENSE                                 170 Bytes
      ▪ workspace
          ▪ .ipynb_checkpoints
              ▪ README-checkpoint.md             1.2 kB
              ▪ wt_quickstart-checkpoint.ipynb 791.4 kB
          ▪ README.md                             1.2 kB
          ▪ apt.txt                               4 Bytes
          ▪ postBuild                            28 Bytes
          ▪ requirements.txt                     17 Bytes
          ▪ wt_quickstart.ipynb                 791.4 kB
  ◦ fetch.txt                                   114 Bytes
  ◦ manifest-md5.txt                            537 Bytes
  ◦ manifest-sha256.txt                         793 Bytes
  ◦ metadata
```

Files (1.6 MB)

| Name | Size |
|---|---|
| 5e7e10163632f4f0c84c51a8.zip | 1.6 MB |

md5:30a0d0e29130f3f3f66b529a08656b89

Show only: Literature (0) Dataset (0) Software (0) Unknown (0)

this version

ecord/515867#collapseTwo

Publication date:
March 27, 2020

DOI:
DOI 10.5072/zenodo.515867

Keyword(s):
Examples Tale

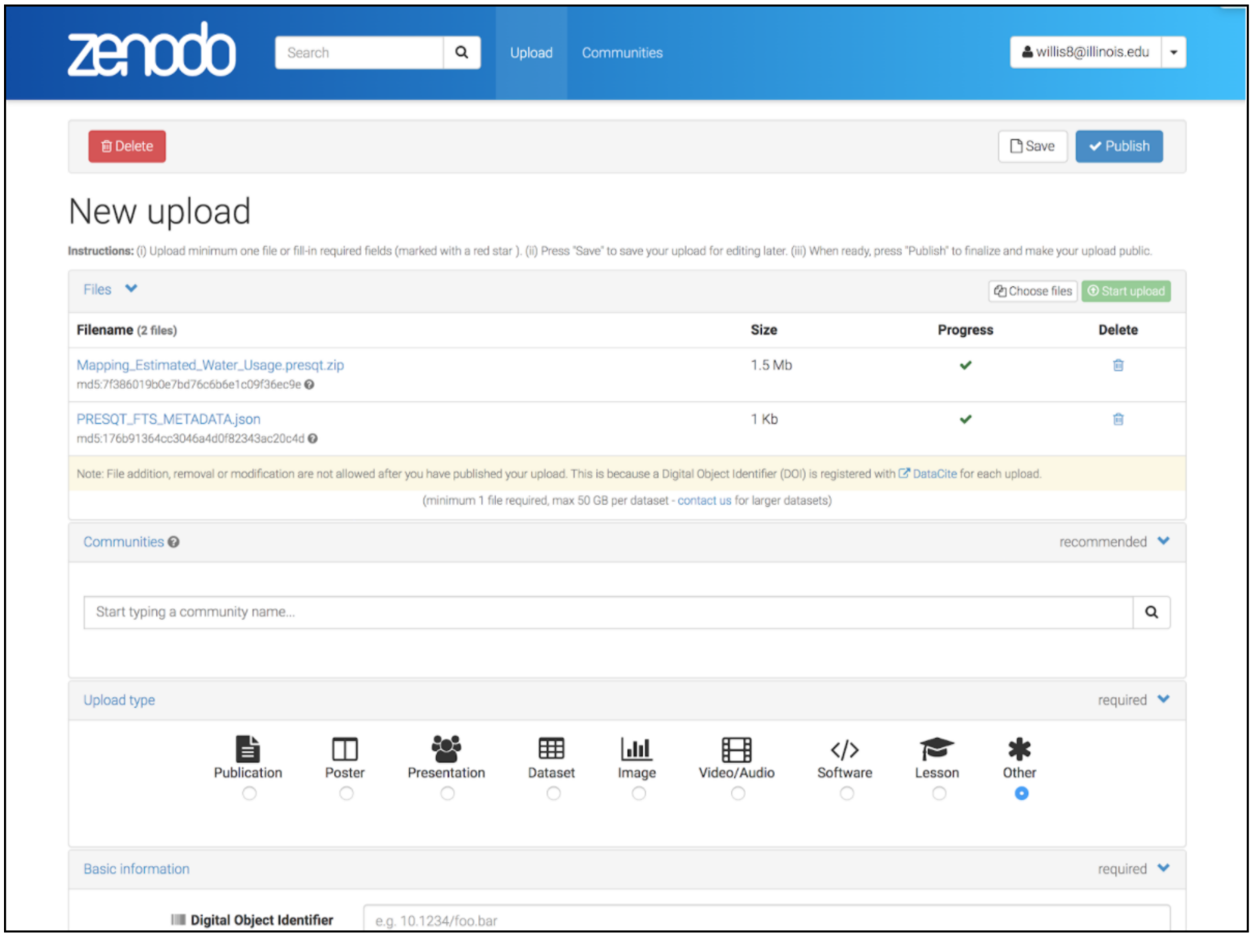
Related identifiers:
Cites
10.5065/D6862DM8
10.5065/D6862DM8

Derived from
10.5072/zenodo.501446
10.5072/zenodo.515858
10.5072/zenodo.502521
10.5072/zenodo.490450

License (for files):
Creative Commons Attribution 4.0 International

| | |
|---|---|
| Version 10 10.5072/zenodo.515867 | Mar 27, 2020 |
| Version 9 10.5072/zenodo.515858 | Mar 27, 2020 |
| Version 8 | Mar 2, 2020 |

Fig. 1: Tale published to Zenodo via WT



New upload

**Instructions:** (i) Upload minimum one file or fill-in required fields (marked with a red star ). (ii) Press "Save" to save your upload for editing later. (iii) When ready, press "Publish" to finalize and make your upload public.

Files

| Filename (2 files) | Size | Progress | Delete |
|---|---|---|---|
| Mapping_Estimated_Water_Usage.presqt.zip md5:7f386019b0e7bd76c6b6e1c09f36ec9e | 1.5 Mb | ✓ | |
| PRESQT_FTS_METADATA.json md5:176b91364cc3046a4d0f82343ac20c4d | 1 Kb | ✓ | |

Note: File addition, removal or modification are not allowed after you have published your upload. This is because a Digital Object Identifier (DOI) is registered with DataCite for each upload.

(minimum 1 file required, max 50 GB per dataset - contact us for larger datasets)

Communities — recommended

Start typing a community name...

Upload type — required

Publication [ ] Poster [ ] Presentation [ ] Dataset [ ] Image [ ] Video/Audio [ ] Software [ ] Lesson [ ] [✓ Other]

Basic information — required

Digital Object Identifier: e.g. 10.1234/foo.bar

Fig. 2: Tale published to Zenodo via PresQT

```
{
    "allKeywords": [],
    "actions": [],
    "extra_metadata": {
        "title": "str",
        "creators": [
            {
                "first_name": "Example",
                "last_name": "User",
                "ORCID": "0931234123"
            }
        ],
        "publication_date": "2021-02-19",
        "description": "This is it.",
        "

        s": [],
        "license": "MIT",
        "related_identifiers": [],
        "references": "Nothing here.",
        "notes": "Nope."
    }
}
```

CHAPTER

# FOURTEEN

# UNDER DEVELOPMENT

CHAPTER

# FIFTEEN

# INDICES

- genindex
- modindex
- search

# HTTP ROUTING TABLE

## /api_v1