Data Tracker

SciLifeLab Data Centre

Mar 12, 2021

CONTENTS

CHAPTER

ONE

API

1.1 Version 1

Base URL for the API is `<url>/api/v1/`. All API description have the base implied before the first `/`.

1.1.1 Order

Note: Only for users with `ORDERS` or `DATA_MANAGEMENT`.

`/order/`

GET

- Get a list of all orders where the user is `editor`.
- All orders will be listed for a user with `DATA_MANAGEMENT`.

POST

- Add a new order.
- Returns the `uuid` of the added order.

`/order/<uuid>/`

GET

- Get information about the order `uuid`.

DELETE

- Delete the order `uuid`.

PATCH

- Update the order `uuid`.

`/order/<uuid>/dataset/`

POST

- Add a new dataset for the order `uuid`.
- Returns the `uuid` of the added dataset.

`/order/<uuid>/log/`

GET

- Get a list of changes for the order `uuid`.

1.1.2 Dataset

`/dataset/`

GET

- Get a list of all datasets.

`/dataset/<uuid>/`

GET

- Get information about the dataset `uuid`.

DELETE

- Delete the dataset `uuid`.

PATCH

- Update the dataset `uuid`.

`/dataset/<uuid>/log/`

GET

- Get a list of changes done to the dataset `uuid`.

1.1.3 Collection

`/collection/`

GET

- Get a list of all collections.

POST

- Add a new collection.

`/collection/<uuid>/`

GET

- Get information about the collection `uuid`.

DELETE

- Delete the collection `uuid`.

PATCH

- Update the collection `uuid`.

`/collection/<uuid>/log/`

GET

- Get a list of changes done to the collection `uuid`.

1.1.4 User

Current User

`/user/me/`

GET

- Get information about the current user.

PATCH

- Update information for the current user.

`/user/me/apikey/`

POST

- Generate a new API key for the current user.
- The new API key is returned.

`/user/me/log/`

GET

- Get a list of changes done to the current user.

`/user/me/actions/`

GET

- Get a list of changes done by the current user.

`/user/me/orders/`

GET

- Get a list of orders where the current user is listed as `editor`.

`/user/me/datasets/`

GET

- Get a list of datasets where the current user is listed as `editor`.

`/user/me/collections/`

GET

- Get a list of collections where the current user is listed as `editor`.

Look Up Users

Note: Only for users with `USER_MANAGEMENT`, or in some cases `USER_SEARCH`.

`/user/`

Note: Only for users with `USER_SEARCH` or `USER_MANAGEMENT`.

GET

- Get a list of all users.
- Users with `USER_SEARCH` will get a limited set of fields.

POST

- Add a new user.

`/user/<uuid>/`

GET

- Get information about the user `uuid`.

PATCH

- Update information about the user `uuid`.

DELETE

- Delete the user `uuid`.

`/user/<uuid>/apikey/`

POST

- Generate a new API key for the user `uuid`.
- The new API key is returned.

`/user/<uuid>/log/`

GET

- Get a list of changes done to the user `uuid`.

`/user/<uuid>/actions/`

GET

- Get a list of changes done by the user with `uuid`.

`/user/<uuid>/orders/`

GET

- Get a list of orders where the user `uuid` is listed as `editor`.

`/user/<uuid>/datasets/`

GET

- Get a list of datasets where the user `uuid` is listed as `editor`.

`/user/<uuid>/collections/`

GET

- Get a list of collections where the user `uuid` is listed as `editor`.

Log In/Log Out

`/logout/`

GET

- Log out the current user.

`/login/oidc/<auth_name>/login/`

GET

- Log in using OpenID Connect (e.g. Elixir AAI) for service `auth_name`.

`/login/oidc/<auth_name>/authorize/`

GET

- Authorize using OpenID Connect (e.g. Elixir AAI) for service `auth_name` (via `login`).

`/login/apikey/`

GET

- Log in using `auth_id` + `api_key`.

CHAPTER

TWO

DATA STRUCTURE

The Data Tracker is based on a few main components:

- Order
- Dataset
- Collection
- User
- Log

2.1 General

- `Title` may never be empty.

2.2 Terminology

- Fields:
  - Fields in the documents for the datatype/collection.
- Computed fields:
  - Values that are either calculated or retrieved from documents in other collection(s).
  - Included when the entity is requested via API.

2.3 Order

- Requires special permission to add (`ORDERS_SPECIAL`)
- May only be accessed and modified by users listed in `editors` or users with `DATA_MANAGEMENT`.
- Can have any number of associated datasets.
- Deleting an order will delete all owned datasets.

2.3.1 Summary

Field	Description	Default	Public
_id	UUID of the Entry	Set by system	Hidden
title	Title of the Entry	Must be non-empty	Hidden
description	Description in markdown	Empty	Hidden
generators	List of users who generated data	Entry creator	Visible (via dataset)
authors	List of users responsible for e.g. samples (e.g PI)	Entry creator	Visible (via dataset)
organisation	User who is data controller	Entry creator	Visible (via dataset)
editors	List of users who can edit the order and datasets	Entry creator	Hidden
datasets	List of associated datasets	Empty	Visible (via dataset)
tags_standard	Tags defined in the system	Empty	Hidden
tags_user	Tags defined by the users	Empty	Hidden

2.3.2 Fields

_id

- UUID of the entry.
- Set by the system upon entry creation, never modified.

title

- Entry title.
- Must be non-empty.

description

- Entry description.
- May use markdown for formatting.
- **Default:** Empty

generators

- List of `users`.
- Corresponds to e.g. the facility or people generating the data (from samples).
- May be shown openly on all associated datasets.
  - Access may be limited by other settings.
- **Default:** The user that created the entry.

authors

- List of `users`.
- Corresponds to e.g. the researcher who leads the project the samples came from.
- May be shown openly on all associated datasets.
  - Access may be limited by other settings.
- **Default:** The user that created the entry.

organisation

- A single `user` who is the data controller for the datasets generated from the order (e.g. a University).
- **Default:** The user that created the entry.

editors

- List of `users`.
- Users that may edit the order and dataset entries. May add datasets to an order.
- **Default:** The user that created the entry.

datasets

- List of datasets associated to the order.
- Cannot be modified directly but must be modified through specialised means.
- **Default:** Empty

tags_standard

- A standard set of tags that are defined by the system.
- **Default:** Empty

tags_user

- User-defined tags for the system.
- **Default:** Empty

2.4 Dataset

- Dataset generated by e.g. facility.
- A dataset must be associated with **one** order.
- Multiple datasets may be associated with the same order.
- The association to a specific order cannot be changed.
  - Once associated with an order, it will stay so.
- Can have identifier(s) (e.g. DOIs).
- Will use some fields from its order:
  - `generators`
  - `authors`
  - `organisation`
  - `editors`

2.4.1 Summary

Field	Description	Default	Public
_id	UUID of the Entry	Set by system	Visible
title	Title of the Entry	Must be non-empty	Visible
description	Description in markdown	Empty	Visible
tags_standard	Tags defined in the system	Empty	Visible
tags_user	Tags defined by the users	Empty	Visible
cross_references	External identifiers, links etc.	Empty	Visible

2.4.2 Fields

_id

- UUID of the entry.
- Set by the system upon entry creation, never modified.

title

- Entry title.
- Must be non-empty.

description

- Entry description.
- May use markdown for formatting.
- **Default:** Empty

tags_standard

- A standard set of tags that are defined by the system.
- **Default:** Empty

tags_user

- User-defined tags for the system.
- **Default:** Empty

cross_references

- External references to the data.
- E.g. DOIs or database IDs.
- **Default:** Empty

2.4.3 Computed fields

related

- `datasets` from order, except the current dataset.

collections

- List of collections containing the current dataset in `datasets`.

generators

- `generators` from order.

authors

- `authors` from order.

organisation

- `organisation` from order.

2.5 Collection

- May be created by any users.
- Can have multiple `editors`.
- Can have identifiers.
- Provides a way of grouping datasets before publication.
- Should aid requesting a DOI from Figshare for the collection.

2.5.1 Summary

Field	Description	Default	Public
_id	UUID of the Entry	Set by system	Visible
title	Title of the Entry	Must be non-empty	Visible
datasets	The associated datasets	Empty	Visible
description	Description in markdown	Empty	Visible
tags_standard	Tags defined in the system	Empty	Visible
tags_user	Tags defined by the users	Empty	Visible
cross_references	External identifiers, links etc.	Empty	Visible
editors	List of users who can edit the collection	Entry creator	Hidden

2.5.2 Fields

_id

- UUID of the collection.
- Set by the system upon entry creation, never modified.

title

- Entry title.

- Must be non-empty.

description

- Entry description.
- May use markdown for formatting.
- **Default:** Empty

tags_standard

- A standard set of tags that are defined by the system.
- **Default:** Empty

tags_user

- User-defined tags for the system.
- **Default:** Empty

cross_references

- External references to the data.
- E.g. DOIs or database IDs.
- **Default:** Empty

editors

- List of `users`.
- Users that may edit the collection.
  - May add datasets to an order.
- **Default:** The user that created the entry.

2.6 User

- Everyone using the system is a user.
  - Including facilities, organisations . . .
- Login via e.g. Elixir AAI or API key.
  - On first login, the user will be added to db.
- API can also be accessed using an API key.
  - API key may be generated by any user.
- A user with the permission `USER_MANAGEMENT` can create and modify users.
- A user with the permission `ORDER_USERS` can create and modify “partial” users.

2.6.1 Summary

Field	Description	Default	Public
_id	UUID of the Entry	Set by system	Hidden
affiliation	User affiliation (e.g. university)	Empty	Visible
api_key	Hash for the API key	Empty	Hidden
api_salt	Salt for API api_key	Empty	Hidden
auth_ids	List of identfiers from e.g. Elixir	Empty	Hidden
email	Email address for the user	Must be non-empty	Hidden
email_public	Email address to show publicly	Empty	Visible
name	Name of the user	Must be non-empty	Visible
orcid	ORCID of the user	Empty	Visible
permissions	List of permissions for the user	Empty	Hidden
url	URL to e.g. homepage	Empty	Visible

2.6.2 Fields

_id

- UUID of the entry.
- Set by the system upon entry creation, never modified.

affiliation

- Affiliation of the user.

api_key

- Hash for the API key for authorization to API or login.

api_salt

- Salt for the API key.

auth_ids

- List of identifiers used by e.g. Elixir AAI.
- Saved as strings.
- The general form is `email@location.suffix::source`, but the style may vary between sources.
- Any of the auth_id can be used with the API key.

email

- Email address for the user.
- **Default:** Must be set

email_public

- Email to show to public on e.g. generated datasets.
- **Default:** Empty.

name

- Name of the user.
  - Could also be name of e.g. facility or university.

orcid

- ORCID of the user.

permissions

- A list of the extra permissions the user has (see *Permissions*).

url

- Url to e.g. a homepage
- If set, it must start with `http://` or `https://`.
- **Default:** Empty

2.7 Log

- Whenever an entry (`order`, `dataset`, `collection`, or `user`) is changed, a log should be written.
- Only visible to entry owners and admins.
- All logs are in the same collection.
- The log needs parsing to show changes between different versions of an entry.
- A full cope of the new entry is saved.
  - In case of deletion, `_id` is saved as `data`.

2.7.1 Summary

Field	Description	Default
_id	UUID of the Entry	Set by system
action	type of action	Must be non-empty
comment	Short description of the action	Empty
data_type	The modified collection (e.g. order)	Must be non-empty
data	Complete copy of the new entry	Must be non-empty
timestamp	Timestamp for the change	Must be non-empty
user	UUID for the user who performed the action	Must be non-empty

2.7.2 Fields

_id

- UUID of the entry.
- Set by the system upon entry creation, never modified.

action

- Type of action
  - Add
  - Edit
  - Delete

comment

- Short description of why it was made
  - "Add Dataset from order `X`".

data_type

- The collection that was modified, e.g. `order`

data

- Add/edit: full copy of the new/updated document.
- Delete: the `_id` of the document.

timestamp

- The time the action was performed.

user

- `_id` of the user that performed the action.
- Can be set to `system` for automated actions (e.g. creating a user after OIDC login)

CHAPTER

THREE

IMPLEMENTATION

3.1 Permissions

- Permissions are managed by topics.
- A user may have multiple topics.
- The topics are defined in `user.py`.
- The topics are defined as a dict:

```
{
  'ENTRY': ('ENTRY', 'ENTRY2'),
  ...
}
```

- Each topic is defined as key, and any other topics that are considered to cover the same task is included as value. - Allows the use a single topic to require permission for an API endpoint.
- `permission_required` is used to check whether a user has the required permission. - It is not defined as a decorator, as it may sometimes need to coexist with an ownership check. - At the beginning of a request, run e.g. `user.permission_required('OWNERS_READ')`.

3.1.1 Current units

LOGGED_IN Task require a logged in user (e.g. show user info). Use the decorator `user.login_required`.

DATA_MANAGEMENT May modify any order, dataset, or project. Includes `ORDERS` and `OWNERS_READ`.

ORDERS May create, edit, and delete orders if listed as an editor for the order. Includes *USER_ADD* and *USER_SEARCH*.

OWNERS_READ May access all entity owner information.

USER_ADD May add users.

USER_SEARCH May list and search for users.

USER_MANAGEMENT May modify any user. Includes *USER_ADD* and *USER_SEARCH*.

3.2 CSRF

A csrf cookie with the name `_csrf_token` is set the first time a request is made to the system. It must be included with the header `X-CSRFToken` for any non-`GET` request.

All cookies are deleted upon logout.

3.3 Testing

All tests are available at `backend/tests`.

3.4 API Keys

The keys are generated using `secrets.token_hex(48)`.

Include a 8-byte randomized salt when calculating hash.

Store the token using `hashlib.sha512(token).hexdigest()`.

CHAPTER FOUR

DEVELOPMENT

4.1 System for development

4.1.1 Prepare config file

Prepare a `config.yaml` file. Just renaming `config.yaml.sample` to `config.yaml` and setting the two `dev` variables to true should be enough.

4.1.2 Build and activate the containers

```
docker-compose up
```

The system can be accessed in a web browser at `localhost:5000`.

4.1.3 Add test data

Set a virtual Python environment, install modules.

```
python -m venv venv
. venv/bin/activate
pip install -r test/requirements.txt
pip install -r backend/requirements.txt
```

Randomized test data can be generated by `test/gen_test_db.py`. Run it using e.g.:

```
PYTHONPATH=backend python3 test/gen_test_db.py
```

4.2 Code reference

4.2.1 app.py

4.2.2 config.py

4.2.3 dataset.py

4.2.4 developer.py

4.2.5 order.py

4.2.6 collection.py

4.2.7 structure.py

4.2.8 user.py

4.2.9 utils.py

4.2.10 validate.py

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search