djehuty

The 4TU.ResearchData and Nikhef repository system

https://github.com/4TUResearchData/djehuty

version 25.1, February 3, 2025

Contents

List of Figures

Chapter 1

Introduction

`djehuty` is the data repository system developed by 4TU.ResearchData and Nikhef. The name finds its inspiration in Thoth, the Egyptian entity that introduced the idea of writing.

1.1 Obtaining the source code

The source code can be downloaded at the Releases[1] page. Make sure to download the `djehuty-25.1.tar.gz` file.

Or, directly download the tarball using the command-line:

```
curl -LO https://github.com/4TUResearchData/djehuty/releases/\
    download/v25.1/djehuty-25.1.tar.gz
```

After obtaining the tarball, it can be unpacked using the `tar` command:

```
tar zxvf djehuty-25.1.tar.gz
```

1.2 Installing the prerequisites

The `djehuty` program needs Python (version 3.9 or higher) and Git to be installed. Additionally, a couple of Python packages need to be installed. The following sections describe installing the prerequisites on various GNU/Linux distributions. To put the software in the context of its environment, figure 1.1 displays the complete run-time dependencies from `djehuty` to `glibc`.

[1] https://github.com/4TUResearchData/djehuty/releases

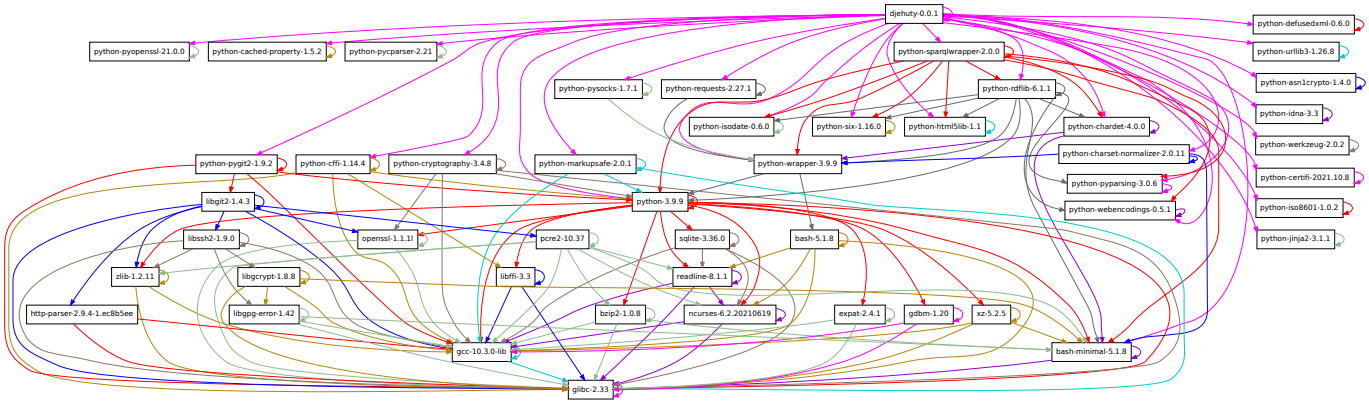

Figure 1.1: *Run-time references when constructed with the packages from GNU Guix.*

The web service of `djehuty` stores its information in a SPARQL 1.1 ("SPARQL 1.1 Overview", 2013) endpoint. We recommend either Blazegraph[2] or Virtuoso open-source edition[3].

1.3 Installation instructions

After obtaining the source code (see section 1.1 'Obtaining the source code') and installing the required tools (see section 1.2 'Installing the prerequisites'), building involves running the following commands:

```
cd djehuty-25.1
autoreconf -vif # Only needed if the "./configure" step does not work.
./configure
make
make install
```

To run the `make install` command, super user privileges may be required. Specify a `--prefix` to the `configure` script to install the tools to a user-writeable location to avoid needing super user privileges.

After installation, the `djehuty` program will be available.

1.4 Pre-built containers

4TU.ResearchData provides Docker container images as a convenience service for each monthly `djehuty` release. The following table outlines the meaning of each image provided. The images are published to Docker Hub[4].

[2] https://blazegraph.com/
[3] http://vos.openlinksw.com/owiki/wiki/VOS
[4] https://hub.docker.com/r/4turesearchdata/djehuty

Image tag	Description
`devel`	Image meant for development purposes. Before it executes the `djehuty` command it checks out the latest codebase. So re-running the same container image may result in running a different version of `djehuty`.
`latest`	This image points to the latest `djehuty` release. It does not automatically update the `djehuty` codebase.
`XX.X`	4TU.ResearchData releases a version each month where the number before the dot refers to the year and the number after the dot refers to the month. Use a specific version image when you want to upgrade at your own pace.

To build the container images for yourself, see the build instructions in the ‘`docker/Dockerfile`’ file.

1.5 RPM packages

4TU.ResearchData provides RPM packages built for Enterprise Linux 9. This RPM depends on packages in the Extra Packages for Enterprise Linux (EPEL) repository.

Filename	Description
djehuty-25.1-1.el9.noarch.rpm	Binary RPM, to install and run `djehuty`.
djehuty-25.1-1.el9.src.rpm	Source RPM, to (re)build from source code.

Chapter 2

Configuring djehuty

Now that djehuty is installed, it's a good moment to look into its run-time configuration options. All configuration can be done through a configuration file, for which an example is available at 'etc/djehuty/djehuty-exa

2.1 Essential options

Option	Description
`bind-address`	The address to bind a TCP socket on.
`port`	The port to bind a TCP socket on.
`alternative-port`	A fall-back port to bind on when `port` is already in use.
`base-url`	The URL on which the instance will be available to the outside world.
`allow-crawlers`	Set to 1 to allow crawlers in the `robots.txt`, otherwise set to 0.
`production`	Performs extra checks before starting. Enable this when running a production instance.
`live-reload`	When set to 1, it reloads Python code on-the-fly. We recommend to set it to 0 when running in production.
`debug-mode`	When set to 1, it will display backtraces and error messages in the web browser. When set to 0, it will only show backtraces and error messages in the web browser.
`use-x-forwarded-for`	When running `djehuty` behind a reverse-proxy server, use the HTTP header `X-Forwarded-For` to log IP address information.
`disable-collaboration`	When set to 1, it disables the "collaborators" feature.
`allowed-depositing-domains`	When unset, any authenticated user may deposit data. Otherwise, this option limits the ability to deposit to users with an e-mail address of the listed domain names.
`cache-root`	`djehuty` can cache query responses to lower the load on the database server. Specify the directory where to store cache files. This element takes an attribute `clear-on-start`, and when set to 1, it will remove all cache files on start-up of `djehuty`.
`profile-images-root`	Users can upload a profile image in `djehuty`. This option should point to a filesystem directory where these profile images can be stored.
`disable-2fa`	Accounts with privileges receive a code by e-mail as a second factor when logging in. Setting this option to 1 disables the second factor authentication.
`sandbox-message`	Display a message on the top of every page.
`notice-message`	Display a message on the main page.
`maintenance-mode`	When set to 1, all HTTP requests result in the displayment of a maintenance message. Use this option while backing up the database, or when performing major updates.

2.2 Configuring the Database

The `djehuty` program stores its state in a SPARQL 1.1 compliant RDF store. Configuring the connection details is done in the `rdf-store` node.

Option	Description
`state-graph`	The graph name to store triplets in.
`sparql-uri`	The URI at which the SPARQL 1.1 endpoint can be reached. When the `sparql-uri` begins with `bdb://`, followed by a path to a filesystem directory, it will use the BerkeleyDB back-end, for which the `berkeleydb` Python package needs to be installed.
`sparql-update-uri`	The URI at which the SPARQL 1.1 Update endpoint can be reached (in case it is different from the `sparql-uri`.

2.3 Audit trails and database reconstruction

The `djehuty` program can keep an audit log of all database modifications made by itself from which a database state can be reconstructed. Whether `djehuty` keeps such an audit log can be configured with the following option:

Option	Description
`enable-query-audit-log`	When set to 1, it writes every SPARQL query that modifies the database in the web logs. This can be replayed to reconstruct the database at a later time. Setting this option to 0 disables this feature. This element takes an attribute `transactions-directory` that should specify an empty directory to which transactions can be written that are extracted from the audit log.

2.3.1 Reconstructing the database from the query audit log

Each query that modifies the database state while the query audit logs are enabled can be extracted from the query audit log using the `--extract-transactions-from-log` command-line option. A timestamp to specify the starting point to extract from can be specified as an argument. The following example displays its use:

```
djehuty web --config-file=config.xml --extract-transactions-from-log="YYYY-MM-DD
   HH:MM:SS"
```

This will create a file for each query in the folder specified in the `transactions-directory` attribute.

To replay the extracted transactions, use the `apply-transactions` command-line option:

```
djehuty web --config-file=config.xml --apply-transactions
```

When a query cannot be executed, the command stops, allowing to fix or remove the query to-be-replayed. Invoking the `--apply-transactions` command a second time will continue replaying where the previous run stopped.

2.4 Configuring storage

Storage locations can be configured with the `storage` node. When configuring multiple locations, `djehuty` attempts to find a file by looking at the first configured location, and in case it cannot find the file there, it will look at the second configured location, and so on, until it has tried each storage location.

This allows for moving files between storage systems transparently without requiring specific interactions with `djehuty` other than having the files made available as a POSIX filesystem.

One use-case that suits this mechanism is letting uploads write to fast online storage and later move the uploaded files to a slower but less costly storage.

Option	Description
`location`	A filesystem path to where files are stored. This is a repeatable property.

2.5 Configuring an identity provider

Ideally, `djehuty` makes use of an external identity provider. `djehuty` can use SAML2.0, ORCID, or an internal identity provider (for testing and development purposes only).

This section will outline the configuration options for each identity provision mechanism.

2.5.1 SAML2.0

For SAML 2.0, the configuration can be placed in the `saml` section under `authentication`. That looks as following:

```
<authentication>
  <saml version="2.0">
    <!-- Configuration goes here. -->
  </saml>
</authentication>
```

The options outlined in the remainder of this section should be placed where the example shows `<!- Configuration`

Option	Description
`strict`	When set to 1, SAML responses must be signed. **Never disable 'strict' mode in a production environment.**
`debug`	Increases logging verbosity for SAML-related messages.
`attributes`	In this section the attributes provided by the identity provider can be aligned to the attributes `djehuty` expects.
`service-provider`	The `djehuty` program fulfills the role of service provider. In this section the certificate and service provider metadata can be configured.
`identity-provider`	In this section the certificate and single-sign-on URL of the identity provider can be configured.
`sram`	In this section, SURF Research Access Management-specific attributes can be configured.

The `attributes` configuration

To create account and author records and to authenticate a user, `djehuty` stores information provided by the identity provider. Each identity provider may provide this information using different attributes. Therefore, the translation from attributes used by `djehuty` and attributes given by the identity provider can be configured. The following attributes must be configured.

Option	Description
`first-name`	A user's first name.
`last-name`	A user's last name.
`common-name`	A user's full name.
`email`	A user's e-mail address.
`groups`	The attribute denoting groups.
`group-prefix`	The prefix for each group short name.

As an example, the attributes configuration for SURFConext looks like this:

```
<attributes>
  <first-name>urn:mace:dir:attribute-def:givenName</first-name>
  <last-name>urn:mace:dir:attribute-def:sn</last-name>
  <common-name>urn:mace:dir:attribute-def:cn</common-name>
  <email>urn:mace:dir:attribute-def:mail</email>
</attributes>
```

And for SURF Research Access Management (SRAM), the attributes configuration looks like this:

```
<attributes>
  <first-name>urn:oid:2.5.4.42</first-name>
  <last-name>urn:oid:2.5.4.4</last-name>
  <common-name>urn:oid:2.5.4.3</common-name>
```

```
  <email>urn:oid:0.9.2342.19200300.100.1.3</email>
  <groups>urn:oid:1.3.6.1.4.1.5923.1.1.1.7</groups>
  <group-prefix>urn:mace:surf.nl:sram:group:[organisation]:[service]</group-
   prefix>
</attributes>
```

The `sram` configuration

When using SURF Research Access Management (SRAM), `djehuty` can persuade SRAM to send an invitation to anyone inside or outside the institution to join the SRAM collaboration that provides access to the `djehuty` instance. To do so, the following attributes must be configured.

Option	Description
`organization-api-token`	An organization-level API token.
`collaboration-id`	The UUID of the collaboration to invite users to.

The `service-provider` configuration

Option	Description
`x509-certificate`	Contents of the public certificate without whitespacing.
`private-key`	Contents of the private key belonging to the `x509-certificate` to sign messages with.
`metadata`	This section contains metadata that may be displayed by the identity provider to users before authorizing them.
`display-name`	The name to be displayed by the identity provider when authorizing the user to the service.
`url`	The URL to the service.
`description`	Textual description of the service.
`organization`	This section contains metadata to describe the organization behind the service.
`name`	The name of the service provider's organization.
`url`	The URL to the web page of the organization.
`contact`	A repeatable section to list contact persons and their roles within the organization. The role can be configured by setting the `type` attribute.
`first-name`	The first name of the contact person.
`last-name`	The last name of the contact person.
`email`	The e-mail address of the contact person. Note that some identity providers prefer functional e-mail addresses (e.g. support@... instead of jdoe@...).

2.5.2 ORCID

ORCID.org plays a key role in making researchers findable. Its identity provider service can be used by `djehuty` in two ways:

1. As primary identity provider to log in and deposit data;
2. As additional identity provider to couple an author record to its ORCID record.

When another identity provider is configured in addition to ORCID, that identity provider will be used as primary and ORCID will only be used to couple author records to the author's ORCID record.

To configure ORCID, the configuration can be placed in the `orcid` section under `authentication`. That looks as following:

```
<authentication>
  <orcid>
    <!-- Configuration goes here. -->
  </orcid>
</authentication>
```

Then the following parameters can be configured:

Option	Description
`client-id`	The client ID provided by ORCID.
`client-secret`	The client secret provided by ORCID.
`endpoint`	The URL to the ORCID endpoint to use.

2.6 Configuring an e-mail server

On various occassions, `djehuty` will attempt to send an e-mail to either an author, a reviewer or an administrator. To be able to do so, an e-mail server must be configured from which the instance may send e-mails.

The configuration is done under the `email` node, and the following items can be configured:

Option	Description
`server`	Address of the e-mail server without protocol specification.
`port`	The port the e-mail server operates on.
`starttls`	When 1, `djehuty` attempts to use StartTLS.
`username`	The username to authenticate with to the e-mail server.
`password`	The password to authenticate with to the e-mail server.
`from`	The e-mail address used to send e-mail from.
`subject-prefix`	Text to prefix in the subject of all e-mails sent from the instance of `djehuty`. This can be used to distinguish a test instance from a production instance.

2.7 Configuring DOI registration

When publishing a dataset or collection, `djehuty` can register a persistent identifier with DataCite. To enable this feature, configure it under the `datacite` node. The following parameters can be configured:

Option	Description
`api-url`	The URL of the API endpoint of DataCite.
`repository-id`	The repository identifier given by DataCite.
`password`	The password to authenticate with to DataCite.
`prefix`	The DOI prefix to use when registering a DOI.

2.8 Configuring Handle registration

Each uploaded file can be assigned a persistent identifier using the Handle system. To enable this feature, configure it under the `handle` node. The following parameters can be configured:

Option	Description
`url`	The URL of the API endpoint of the Handle system implementor.
`certificate`	Certificate to use for authenticating to the endpoint.
`private-key`	The private key paired with the certificate used to authenticate to the endpoint.
`prefix`	The Handle prefix to use when registering a handle.
`index`	The index to use when registering a handle.

2.9 Configuring IIIF support

When publishing images, `djehuty` can enable the IIIF Image API for the images. It uses `libvips` and `pyvips` under the hood to perform image manipulation. The following parameters can be configured:

Option	Description
`enable-iiif`	Enable support for the IIIF image API. This requires the `pyvips` package to be available in the run-time environment.
`iiif-cache-root`	The directory to store the output of IIIF Image API requests to avoid re-computing the image.

2.10 Customizing looks

With the following options, the instance can be branded as necessary.

Option	Description
`site-name`	Name for the instance used in the title of a browser window and as default value in the publisher field for new datasets.
`site-description`	Description used as a meta-tag in the HTML output.
`site-shorttag`	Used as keyword and as Git remote name.
`support-email-address`	E-mail address used in e-mails sent to users in automated messages.
`custom-logo-path`	Path to a PNG image file that will be used as logo on the website.
`custom-favicon-path`	Path to an ICO file that will be used as favicon.
`small-footer`	HTML that will be used as footer for all pages except for the main page.
`large-footer`	HTML that will be used as footer on the main page.
`show-portal-summary`	When set to 1, it shows the repository summary of number of datasets, authors, collections, files and bytes on the main page.
`show-institutions`	When set to 1, it shows the list of institutions on the main page.
`show-science-categories`	When set to 1, it shows the subjects (categories) on the main page.
`show-latest-datasets`	When set to 1, it shows the list of latest published datasets on the main page.
`colors`	Colors used in the HTML output. See section 2.10.1.

2.10.1 Customizing colors

The following options can be configured in the `colors` section.

Option	Description
`primary-color`	The main background color to use.
`primary-foreground-color`	The main foreground color to use.
`primary-color-hover`	Color to use when hovering a link.
`primary-color-active`	Color to use when a link is clicked.
`privilege-button-color`	The background color of buttons for privileged actions.
`footer-background-color`	Color to use in the footer.

2.11 Configuring privileged users

By default an authenticated user may deposit data. But users can have additional roles; for example: a dataset reviewer, a technical administrator or a quota reviewer.

Such additional roles are configured in terms of privileges. The following privileges can be configured in the `privileges` section:

Option	Description
may-administer	Allows access to perform maintenance tasks, view accounts and view reports on restricted and embargoed datasets.
may-run-sparql-queries	Allows to run arbitrary SPARQL queries on the database.
may-impersonate	Allows to log in to any account and therefore perform any action as that account.
may-review	Allows to see which datasets are sent for review, and allows to perform reviews.
may-review-quotas	Allows access to see requests for storage quota increases and approve or decline them.
may-review-integrity	Allows access to an API call that provides statistics on the accessibility of files on the filesystem.
may-process-feedback	Accounts with this privilege will receive e-mails with the information entered into the feedback form by other users.
may-receive-email-notifications	This "privilege" can be used to disable sending any e-mails to an account by setting it to 0. The default is 1.

To enable a privilege for an account, set the value of the desired privilege to 1. Privileges are disabled by default, except for may-receive-email-notifications which defaults to 1.

```
<privileges>
  <account email="you@example.com" orcid="0000-0000-0000-0001">
    <may-administer>1</may-administer>
    <may-run-sparql-queries>1</may-run-sparql-queries>
    <may-impersonate>1</may-impersonate>
    <may-review>0</may-review>
    <may-review-quotas>0</may-review-quotas>
    <may-review-integrity>0</may-review-integrity>
    <may-process-feedback>0</may-process-feedback>
    <may-receive-email-notifications>1</may-receive-email-notifications>
  </account>
</privileges>
```

Chapter 3

Knowledge graph

Djehuty processes its information using the Resource Description Framework (Lassila, 1999). This chapter describes the parts that make up the data model of `djehuty`.

This chapter dives into the structure of the data model, but does not describe every property. When running an instance of `djehuty`, the "Exploratory" available in the "Admin panel" can be used to explore every property.

3.1 Use of vocabularies

Throughout this chapter, abbreviated references to ontologies are used. Table 3.1 lists these abbreviations.

Abbreviation	Ontology URI
`djht`	Internal and unpublished ontology.
`rdf`	http://www.w3.org/1999/02/22-rdf-syntax-ns#
`rdfs`	http://www.w3.org/2000/01/rdf-schema#
`xsd`	http://www.w3.org/2001/XMLSchema#

Table 3.1: Lookup table for vocabulary URIs and their abbreviations.

3.2 Notational shortcuts

In addition to abbreviating ontologies with their prefix we use another notational shortcut. To effectively communicate the structure of the RDF graph used by `djehuty` we introduce a couple of shorthand notations.

3.2.1 Notation for typed triples

When the `object` in a triple is *typed*, we introduce the shorthand to only show the type, rather than the actual value of the `object`. Figure 3.1 displays this for URIs, and figure 3.2 displays this for literals.

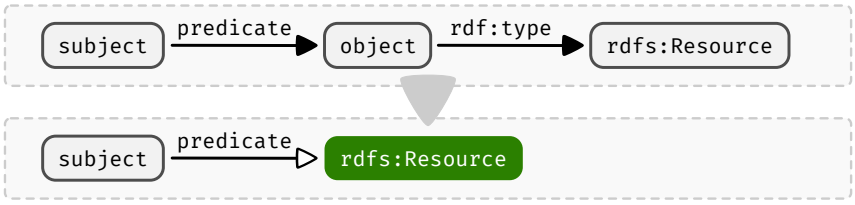

Figure 3.1: *Shorthand notation for triples with an* `rdf:type` *which features a hollow predicate arrow and a colored type specifier with rounded corners.*

Literals are depicted by rectangles (with sharp edges) in contrast to URIs which are depicted as rectangles with rounded edges.

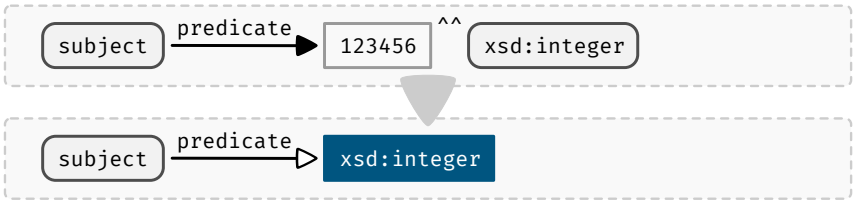

Figure 3.2: *Shorthand notation for triples with a literal, which features a hollow predicate arrow and a colored rectangular type specifier.*

When the subject of a triple is the shorthand type, assume the subject is not the type itself but the subject which has that type.

3.2.2 Notation for `rdf:List`

To preserve the order in which lists were formed, the data model makes use of `rdf:List` with numeric indexes. This pattern will be abbreviated in the remainder of the figures as displayed in figure 3.3.

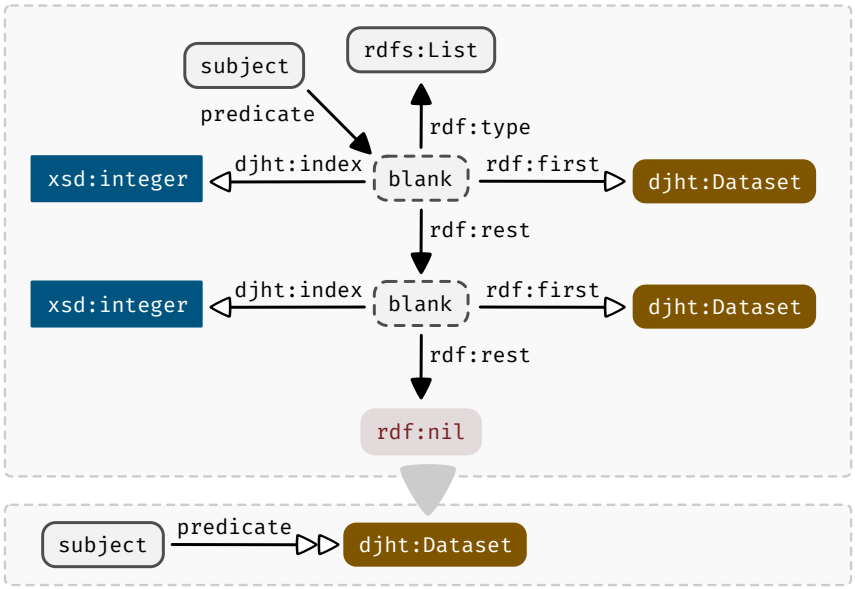

Figure 3.3: *Shorthand notation for `rdf:List` with numeric indexes, which features a hollow double-arrow. Lists have arbitrary lengths, and the numeric indexes use 1-based indexing.*

The hollow double-arrow depicts the use of an `rdf:List` with numeric indexes.

3.3 Datasets

Datasets play a central role in the repository system because every other type links in one way or another to it. The user submits files along with data about those bytes as a single record which we call a `djht:Dataset`. Figure 3.4 shows how the remainder of types in this chapter relate to a `djht:Dataset`.

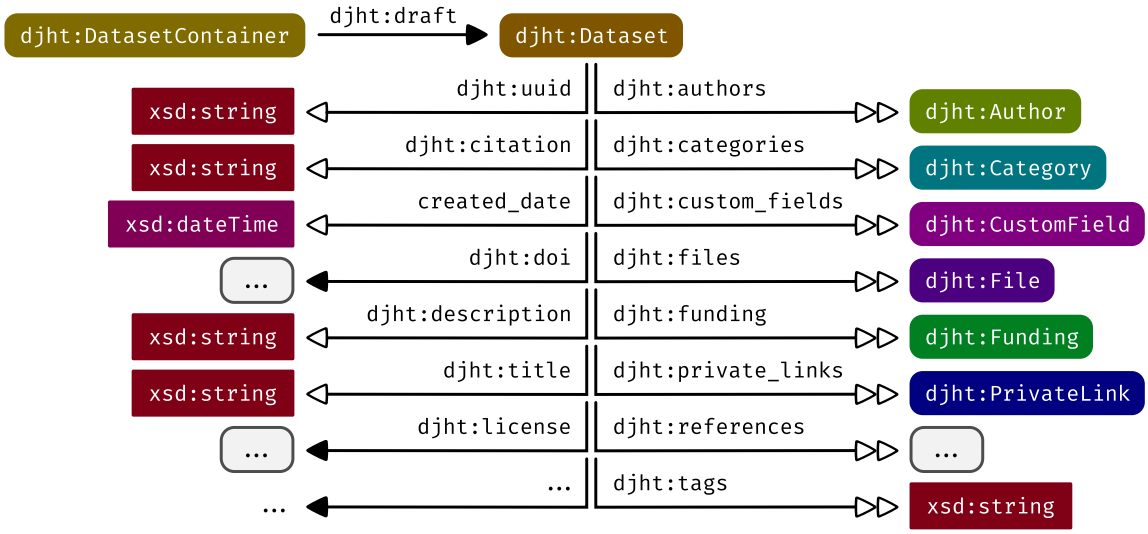

Figure 3.4: *The RDF pattern for a `djht:Dataset`. For a full overview of `djht:Dataset` properties, use the exploratory from the administration panel.*

Datasets are versioned records. The data and metadata between versions can differ, except all versions of a dataset share an identifier. We use `djht:DatasetContainer` to describe the version-unspecific properties of a set of versioned datasets.

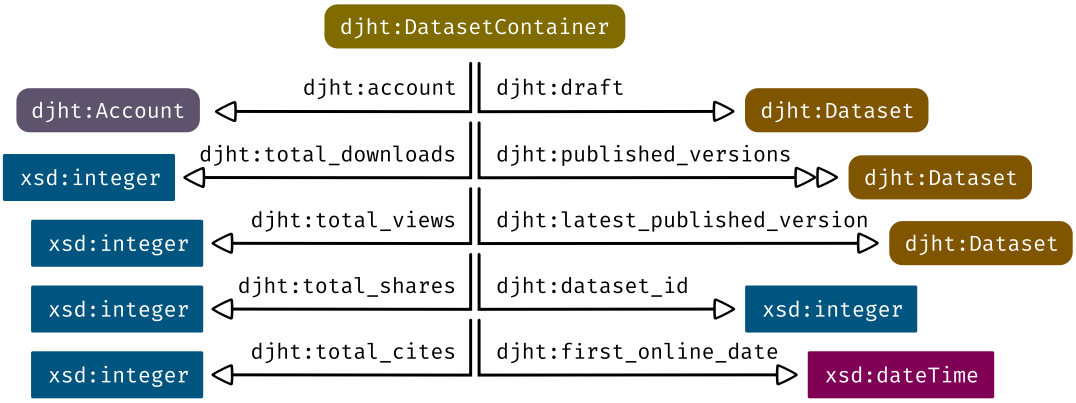

Figure 3.5: *The RDF pattern for a `djht:DatasetContainer`. All versions of a dataset share a `djht:dataset_id` and a UUID in the container URI.*

The data model follows a natural expression of published versions as a linked list. Figure 3.5 further reveals that the *view*, *download*, *share* and *citation* counts are stored in a version-unspecific way.

3.4 Collections

Collections provide a way to group `djht:Dataset` objects.

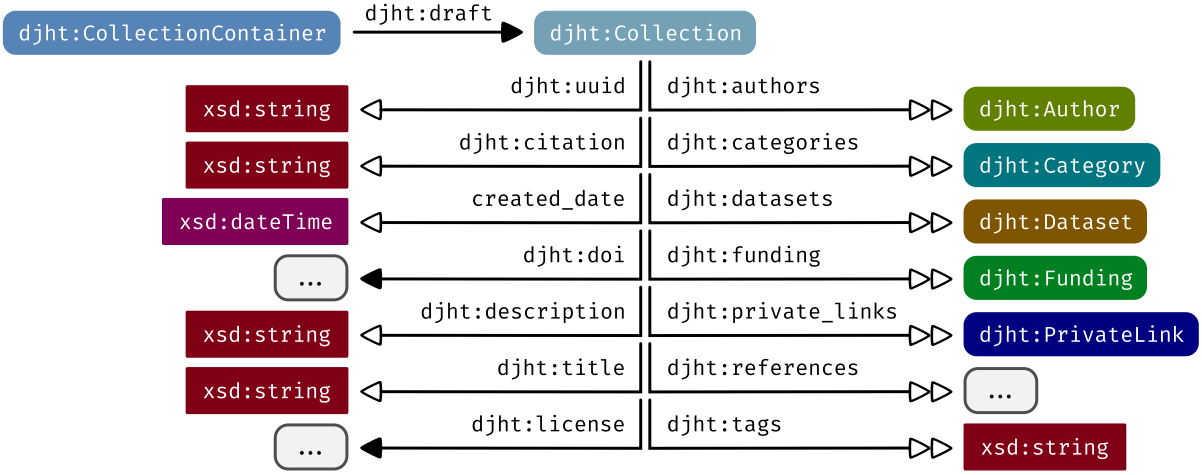

Figure 3.6: *The RDF pattern for a `djht:Collection`. For a full overview of `djht:Collection` properties, use the exploratory from the administration panel.*

Collections are (just like Datasets) versioned records. The metadata between versions can differ, except all versions of a collection share an identifier. We use `djht:CollectionContainer` to describe the version-unspecific properties of a set of versioned collections.

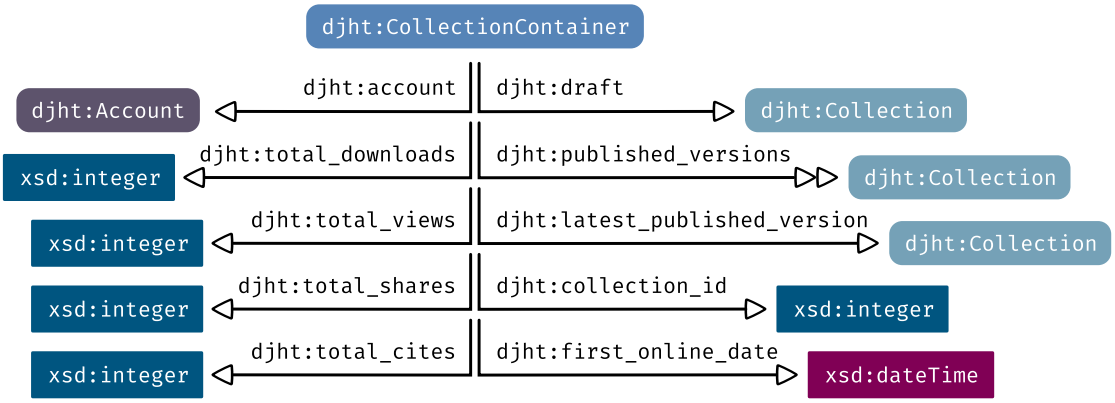

Figure 3.7: *The RDF pattern for a `djht:CollectionContainer`. All versions of a collection share a `djht:collection_id` and a UUID in the container URI.*

The data model follows a natural expression of published versions as a linked list. Figure 3.7 further reveals that the *view*, *download*, *share* and *citation* counts are stored in a version-unspecific way.

3.5 Authors

djehuty keeps records of authors including their full name, ORCID, and e-mail address. Furthermore, each `djht:Account` has a linked `djht:Author` record.

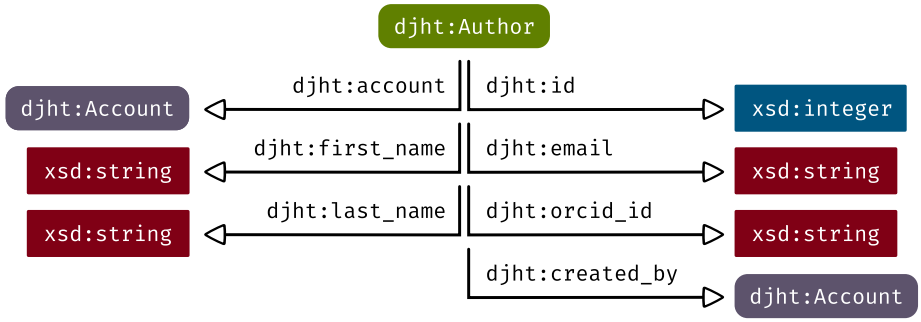

Figure 3.8: *The RDF pattern for an `djht:Author`.*

3.6 Accounts

djehuty uses an external identity provider, but stores an e-mail address, full name, and preferences for categories.

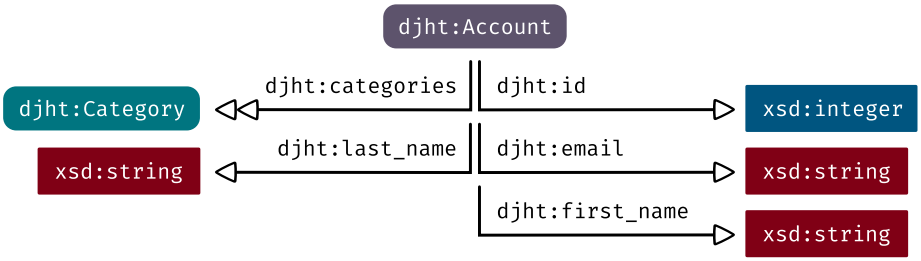

Figure 3.9: *The RDF pattern for an* `djht:Account`.

3.7 Funding

When the `djht:Dataset` originated out of a funded project, the funders can be listed using `djht:Funding`. Figure 3.10 displays the details for this structure.

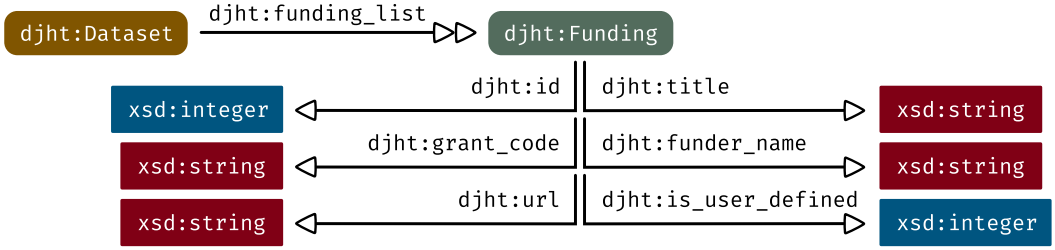

Figure 3.10: *The RDF pattern for a* `djht:Funding`.

3.8 Categories

Categories in `djehuty` are a controlled vocabulary based on the Australian and New Zealand Standard Research Classification (ANZSRC). The hierarchical structure is captured by using `id` and `parent_id` properties.

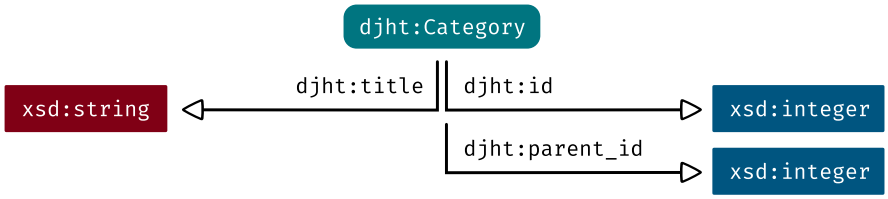

Figure 3.11: *The RDF pattern for an* `djht:Category`.

3.9 Institutions/groups

An `djht:Account` has an affiliation with an institute or research group. The `djht:InstitutionGroup` is stored per `djht:Dataset` and `djht:Collection`. The groups can be structured hierarchically by using the `id` and `parent_id` properties.

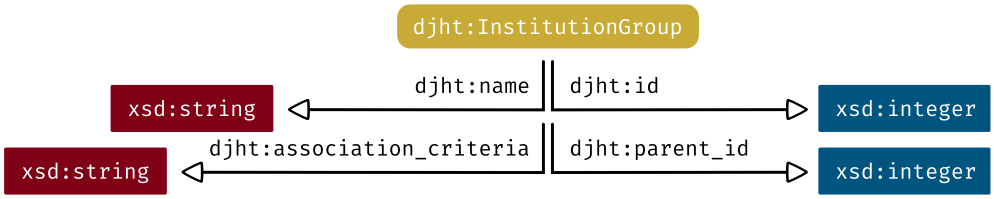

Figure 3.12: *The RDF pattern for an `djht:InstitutionGroup`.*

3.10 Files

A `djht:Dataset` keeps a list of `djht:File` records. The file metadata is stored in the knowledge graph while the file contents are stored on a filesystem. The location of the file data is tracked via the `djht:filesystem_location` property.

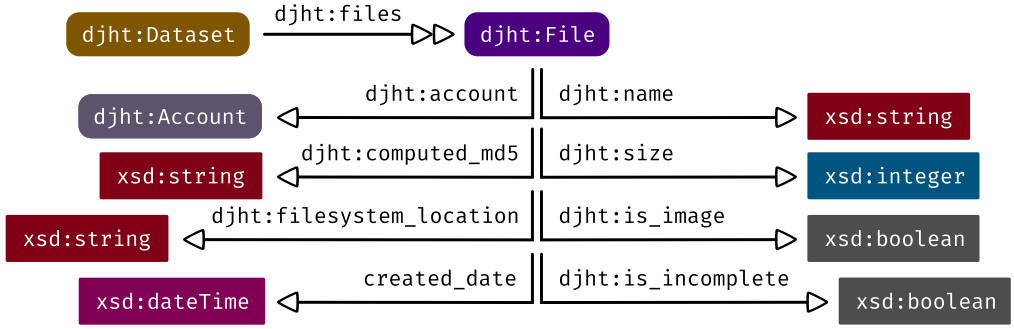

Figure 3.13: *The RDF pattern for a `djht:File`.*

3.11 Private links

Before a `djht:Dataset` or a `djht:Collection` is made publically available, it can be shared using a private link.

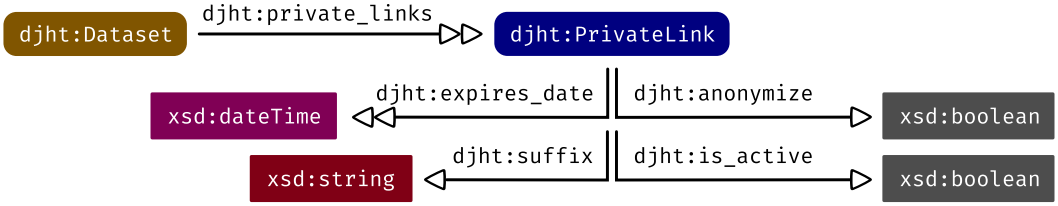

Figure 3.14: *The RDF pattern for a `djht:PrivateLink`.*

The figure 3.14 displays how private links are stored for a `djht:Dataset`, and it works the same for a `djht:Collection`.

Chapter 4

Contributing

This chapter outlines how to set up an instance of djehuty with the goal of modifying its source code. Or in other words: this is the developer setup.

4.1 Setting up a development environment

First, we need to obtain the latest version of the source code:

```
$ git clone https://github.com/4TUResearchData/djehuty.git
```

Next, we need to create a somewhat isolated Python environment:

```
$ python -m venv djehuty-env
$ . djehuty-env/bin/activate
[env]$ cd djehuty
[env]$ pip install -r requirements.txt
```

And finally, we can install djehuty in the virtual environment to make the djehuty command available:

```
[env]$ sed -e 's/@VERSION@/0.0.1/g' pyproject.toml.in > pyproject.toml
[env]$ pip install --editable .
```

If all went well, we will now be able to run djehuty:

```
[env]$ djehuty --help
```

4.2 Configuring djehuty

Invoking djehuty web starts the web interface of djehuty. On what port it makes itself available can be configured in its configuration file. An example of a configuration file can be found in ‘etc/djehuty/djehuty-examp
We will use the example configuration as the basis to configure it for the development environment.

```
[env]$ cp etc/djehuty/djehuty-example-config.xml config.xml
```

In the remainder of the chapter we will assume a value of `127.0.0.1` for `bind-address` and a value of `8080` for `port`.

4.2.1 Modifications to the example configuration for developers

The chapter 2 'Configuring djehuty' describes each configuration option for `djehuty`. The remainder of sections here contain a fast-path through configuring `djehuty` for use in a development setup.

Live reload

The `djehuty` program can be configured to automatically reload itself when a change is detected by setting `live-reload` to `1`.

Configuring authentication with ORCID

The `djehuty` program does not have Identity Provider (IdP) capabilities, so in order to log into the system we must configure an external IdP. With an ORCID account comes the ability to set up an OAuth endpoint. Go to developer-tools at orcid.org. When setting up the OAuth at ORCID, choose `http://127.0.0.1:8080/login` as redirect URI.

Modify the following bits to reflect the settings obtained from ORCID.

```
<authentication>
  <orcid>
    <client-id>APP-XXXXXXXXXXXXXXXX</client-id>
    <client-secret>XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX</client-secret>
    <endpoint>https://orcid.org/oauth</endpoint>
  </orcid>
</authentication>
```

To limit who can log into a development system, accounts are not automatically created for ORCID as IdP. So we need to configure who can log in by creating a record in the `privileges` section of the configuration file.

This is also a good moment to configure additional privileges for your account. In the following snippet, configure the ORCID with which you will log into the system in the `orcid` argument.

```
<privileges>
  <account email="you@example.com" orcid="0000-0000-0000-0001">
    <may-administer>1</may-administer>
    <may-impersonate>1</may-impersonate>
    <may-review>1</may-review>
  </account>
</privileges>
```

4.2.2 Invoking `djehuty`

Once we've configured `djehuty` for development use, we can start the web interface by running:

```
[env]$ djehuty web --initialize --config-file=config.xml
```

The `--initialize` option creates the internal account record and associates the specified ORCID with it. We only need to run `djehuty` with the `--initialize` option once.

By now, we should be able to visit `djehuty` through a web browser at localhost:8080, unless configured differently. We should be able to log in through ORCID, and access all features of `djehuty`.

4.3 Navigating the source code

In this section, we trace the path from invoking `djehuty` to responding to a HTTP request.

4.3.1 Starting point

Because `djehuty` is installable as a Python package, we can find the starting point for running `djehuty` in `pyproject.toml`. It reads:

```
[project.scripts]
djehuty = djehuty.ui:main
```

So, we start our tour at '`src/djehuty/ui.py`' in the procedure called `main`.

4.3.2 How `djehuty` initializes

The `main` procedure calls `main_inner`, which handles the command-line arguments. When invoking `djehuty`, we usually invoke `djehuty web`, which is handled by the following snippet:

```
import djehuty.web.ui as web_ui
...
if args.command == "web":
    web_ui.main (args.config_file, True, args.initialize,
                 args.extract_transactions_from_log,
                 args.apply_transactions)
```

So, the entry-point for the `web` subcommand is found in `src/djehuty/web/ui.py` at the `main` procedure.

This procedure essentially sets up an instance of `WebServer` (found in `src/djehuty/web/wsgi.py` and uses `werkzeug`'s `run_simple` to start the web server.

4.3.3 Translating URI paths to internal procedures

An instance of the `WebServer` is passed along in werkzeug's `run_simple` procedure. Werkzeug calls the instance directly, which is handled by the `__call__` procedure of the `WebServer` class. The `__call__` procedure invokes its `wsgi` instance, which is configured as following:

```
self.wsgi = SharedDataMiddleware(self.__respond, self.static_roots)
```

The `__respond` procedure calls `__dispatch_request`

In `__dispatch_request`, the requested URI is translated into the procedure name using the `url_map`. So, except for static resources in the `src/djehuty/web/resources` folder and pre-configured static pages, URIs are handled by a procedure in the `WebServer` instance.

A mapping between a URI and the procedure that is executed to handle the request to that URI can be found in the `url_map` defined in the `WebServer` class in '`wsgi.py`'.

4.3.4 Diving into the code that displays the homepage

As an example, in the `url_map`, we can find the following line:

```
R("/", self.ui_home),
```

In this case, `self` is a reference to an instance of the `WebServer` class, so we look for a procedure called `ui_home` inside the `WebServer` class. Some code editors have a feature to "go to definition" which helps navigating.

The `ui_home` gathers the summary numbers from the SPARQL endpoint with the following line:

```
summary_data = self.db.repository_statistics()
```

And a list of the latest datasets with the following line:

```
records = self.db.latest_datasets_portal(30)
```

It then passes that information to the `__render_template` procedure which renders the '`portal.html`' in the '`src/djehuty/web/resources/html_templates`' folder. The Jinja[1] package is used to interpret the template.

```
return self.__render_template (request, "portal.html",
                               summary_data = summary_data,
                               latest = records, ...)
```

4.3.5 Database communication

In the `ui_home` procedure, we found a call to the `self.db.repository_statistics` procedure. To find out by hand where that procedure can be found, we can look for the place where `self.db` is assigned a value:

```
self.db = database.SparqlInterface()
```

[1] https://jinja.palletsprojects.com/en/3.1.x/

And from there look up where `database` comes from:

```
from djehuty.web import database
```

From which we can conclude that it can be found in '`src/djehuty/web/database.py`'.

In the `repository_statistics` procedure, we find a call to `self.__query_from_template` followed by a call to `__run_query` which takes the output of the former procedure as its input.

As the name implies, `__run_query` sends the query to the SPARQL endpoint and retrieves the results by putting them in a list of Python dictionaries.

The `self.__query_from_template` procedure takes one parameter, which is the name of the template file (minus the extension) that contains a SPARQL query. These templates can be found in the '`src/djehuty/web/resources/sparql_templates`' folder.

Chapter 5

Application Programming Interface

The application programming interface (API) provided by `djehuty` allows for automating tasks otherwise done through the user interface. In addition to automation, the API can also be used to gather additional information, like statistics on Git repositories.

Throughout this chapter we provide examples for using the API using `curl` and `jq`. Another way of seeing the API in action is to use the developer tools in a web browser while performing the desired action using the web user interface.

5.1 The `/v2` public interface

The `v2` API was designed by Figshare[1]. `djehuty` implements a backward-compatible version of it, with the following differences:

1. The `id` property is superseded by the `uuid` property.

2. Error handling is done through precise HTTP error codes, rather than always returning `400` on a usage error.

Unless specified otherwise, the HTTP `Content-Type` to interact with the API is `application/json`. In the case an API call returns information, don't forget to set the HTTP `Accept` header appropriately.

5.1.1 `/v2/articles` (HTTP `GET`)

This API endpoint can be used to retrieve a list of published datasets. Passing (one of) the following parameters will filter or sort the list of datasets:

[1] https://figshare.com

Parameter	Required	Description
`order`	Optional	Field to use for sorting.
`order_direction`	Optional	Can be either `asc` or `desc`.
`institution`	Optional	The institution identifier to filter on.
`published_since`	Optional	When set, datasets published before this timestamp are dropped from the results.
`modified_since`	Optional	When set, only datasets modified after this timestamp are shown from the results.
`group`	Optional	The group identifier to filter on.
`resource_doi`	Optional	The DOI of the associated journal publication. When set, only returns datasets associated with this DOI.
`item_type`	Optional	Either `3` for datasets or `9` for software.
`doi`	Optional	The DOI of the dataset to search for.
`handle`	Optional	Unused.
`page`	Optional	The page number used in combination with `page_size`.
`page_size`	Optional	The number of datasets per page. Used in combination with `page`.
`limit`	Optional	The maximum number of datasets to output. Used together with `offset`.
`offset`	Optional	The number of datasets to skip in the output. Used together with `limit`.

Example usage:

```
curl "https://data.4tu.nl/v2/articles?limit=100&published_since=2024-07-25" | jq
```

Output of the example:

```
[ /* Example output has been shortened. */
  {
    "id": null,
    "uuid": "4f8a9423-83fc-4263-9bb7-2aa83d73865d",
    "title": "Measurement data of a Low Speed Field Test of Tractor Se...",
    "doi": "10.4121/4f8a9423-83fc-4263-9bb7-2aa83d73865d.v1",
    "handle": null,
    "url": "https://data.4tu.nl/v2/articles/4f8a...865d",
    "published_date": "2024-07-26T10:39:57",
    "thumb": null,
    "defined_type": 3,
    "defined_type_name": "dataset",
    "group_id": 28589,
    "url_private_api": "https://data.4tu.nl/v2/account/articles/4f8a...865d",
    "url_public_api": "https://data.4tu.nl/v2/articles/4f8a...865d",
```

```
    "url_private_html": "https://data.4tu.nl/my/datasets/4f8a...865d/edit",
    "url_public_html": "https://data.4tu.nl/datasets/4f8a...865d/1",
    ...
  }
]
```

5.1.2 /v2/articles/search (HTTP POST)

In addition to the parameters of section 5.1.1 '/v2/articles (HTTP GET)', the following parameters can be used.

Parameter	Required	Description
search_for	Optional	The terms to search for.

Example usage:

```
curl --request POST \
     --header "Content-Type: application/json"\
     --data '{ "search_for": "djehuty" }'\
     https://data.4tu.nl/v2/articles/search | jq
```

Output of the example:

```
[ /* Example output has been shortened. */
  {
    "id": null,
    "uuid": "342efadc-66f8-4e9b-9d27-da7b28b849d2",
    "title": "Source code of the 4TU.ResearchData repository",
    "doi": "10.4121/342efadc-66f8-4e9b-9d27-da7b28b849d2.v1",
    "handle": null,
    "url": "https://data.4tu.nl/v2/articles/342e...49d2",
    "published_date": "2023-03-20T11:29:10",
    "thumb": null,
    "defined_type": 9,
    "defined_type_name": "software",
    "group_id": 28586,
    "url_private_api": "https://data.4tu.nl/v2/account/articles/342e...49d2",
    "url_public_api": "https://data.4tu.nl/v2/articles/342e...49d2",
    "url_private_html": "https://data.4tu.nl/my/datasets/342e...49d2/edit",
    "url_public_html": "https://data.4tu.nl/datasets/342e...49d2/1",
    ...
  }
]
```

5.1.3 /v2/articles/<dataset-id> (HTTP GET)

This API endpoint can be used to retrieve detailed metadata for the dataset identified by `dataset-id`.

Example usage:

```
curl https://data.4tu.nl/v2/articles/342efadc-66f8-4e9b-9d27-da7b28b849d2 | jq
```

Output of the example:

```
{ /* Example output has been shortened. */
  "files": ...,
  "custom_fields": ...,
  "authors": ...,
  "description": "<p>This dataset contains the source code of the 4TU...",
  "license": ...,
  "tags": ...,
  "categories": ...,
  "references": ...,
  "id": null,
  "uuid": "342efadc-66f8-4e9b-9d27-da7b28b849d2",
  "title": "Source code of the 4TU.ResearchData repository",
  "doi": "10.4121/342efadc-66f8-4e9b-9d27-da7b28b849d2.v1",
  "url": "https://data.4tu.nl/v2/articles/342e...49d2",
  "published_date": "2023-03-20T11:29:10",
  "timeline": ...,
  ...
}
```

5.1.4 /v2/articles/<dataset-id>/versions (HTTP GET)

This API endpoint can be used to retrieve a list of versions for the dataset identified by `dataset-id`.

Example usage:

```
curl https://data.4tu.nl/v2/articles/342efadc-66f8-4e9b-9d27-da7b28b849d2/
   versions | jq
```

Output of the example:

```
[
  {
    "version": 1,
    "url": "https://data.4tu.nl/v2/articles/342e...49d2/versions/1"
  }
]
```

5.1.5 /v2/articles/<dataset-id>/versions/<version> (HTTP GET)

This API endpoint can be used to retrieve detailed metadata of the version `version` for the dataset identified by `dataset-id`.

Example usage:

```
curl https://data.4tu.nl/v2/articles/342e...49d2/versions/1 | jq
```

The output of the example is identical to the example output of section 5.1.3 '/v2/articles/<dataset-id> (HTTP GET)'.

5.1.6 /v2/articles/<dataset-id>/versions/<version>/embargo (HTTP GET)

This API endpoint can be used to retrieve embargo information of the version `version` for the dataset identified by `dataset-id`.

Example usage:

```
curl https://data.4tu.nl/v2/articles/c127...8fd7/versions/2/embargo | jq
```

Output of the example:

```
{
  "is_embargoed": true,
  "embargo_date": "2039-06-30",
  "embargo_type": "article",
  "embargo_title": "Under embargo",
  "embargo_reason": "<p>Need consent to publish the data</p>",
  "embargo_options": []
}
```

5.1.7 /v2/articles/<dataset-id>/files (HTTP GET)

This API endpoint can be used to retrieve the list of files associated with the dataset identified by `dataset-id`.

Example usage:

```
curl https://data.4tu.nl/v2/articles/342efadc-66f8-4e9b-9d27-da7b28b849d2/files
```

Output of the example:

```
[ /* Example output has been shortened. */
  {
    "id": null,
    "uuid": "d3e1c325-7fa9-4cb9-884e-0b9cd2059292",
    "name": "djehuty-0.0.1.tar.gz",
```

```
    "size": 3713709,
    "is_link_only": false,
    "is_incomplete": false,
    "download_url": "https://data.4tu.nl/file/342e...49d2/d3e1...9292",
    "supplied_md5": null,
    "computed_md5": "910e9b0f79a0af548f59b3d8a56c3bf4"
  }
]
```

5.1.8 /v2/articles/<dataset-id>/files/<file-id> (HTTP GET)

This API endpoint can be used to retrieve all metadata of the file identified by `file-id` associated with the dataset identified by `dataset-id`.

Example usage:

```
curl https://data.4tu.nl/v2/articles/342e...49d2/files/d3e1...9292 | jq
```

Output of the example:

```
{ /* Example output has been shortened. */
  "id": null,
  "uuid": "d3e1c325-7fa9-4cb9-884e-0b9cd2059292",
  "name": "djehuty-0.0.1.tar.gz",
  "size": 3713709,
  "is_link_only": false,
  "is_incomplete": false,
  "download_url": "https://data.4tu.nl/file/342e...49d2/d3e1...9292",
  "supplied_md5": null,
  "computed_md5": "910e9b0f79a0af548f59b3d8a56c3bf4"
}
```

5.1.9 /v2/collections (HTTP GET)

This API endpoint can be used to retrieve a list of collections published in the data repository.

The following parameters can be used:

Parameter	Required	Description
`order`	Optional	Field to use for sorting.
`order_direction`	Optional	Can be either `asc` or `desc`.
`institution`	Optional	The institution identifier to filter on.
`published_since`	Optional	When set, collections published before this timestamp are dropped from the results.
`modified_since`	Optional	When set, only collections modified after this timestamp are shown from the results.
`group`	Optional	The group identifier to filter on.
`resource_doi`	Optional	The DOI of the associated journal publication. When set, only returns collections associated with this DOI.
`doi`	Optional	The DOI of the collection to search for.
`handle`	Optional	Unused.
`page`	Optional	The page number used in combination with `page_size`.
`page_size`	Optional	The number of collections per page. Used in combination with `page`.
`limit`	Optional	The maximum number of collections to output. Used together with `offset`.
`offset`	Optional	The number of collections to skip in the output. Used together with `limit`.

Example usage:

```
curl "https://data.4tu.nl/v2/collections?limit=100&published_since=2024-07-25" |
   jq
```

Output of the example:

```
[ /* Example output has been shortened. */
  {
    "id": null,
    "uuid": "0fe9ab80-6e6a-4087-a509-ce09dddfa3d9",
    "title": "PhD research 'Untangling the complexity of local water ...'",
    "doi": "10.4121/0fe9ab80-6e6a-4087-a509-ce09dddfa3d9.v1",
    "handle": "",
    "url": "https://data.4tu.nl/v2/collections/0fe9...fa3d9",
    "timeline": {
      "posted": "2024-08-13T14:09:52",
      "firstOnline": "2024-08-13T14:09:51",
      ...
    },
    "published_date": "2024-08-13T14:09:52"
  },
  ...
```

```
]
```

5.1.10 /v2/collections/search (HTTP POST)

This API endpoint can be used to search for collections published in the data repository.

In addition to the parameters of section 5.1.9 '/v2/collections (HTTP GET)', the following parameters can be used.

Parameter	Required	Description
search_for	Optional	The terms to search for.

Example usage:

```
curl --request POST \
     --header "Content-Type: application/json"\
     --data '{ "search_for": "wingtips" }'\
     https://data.4tu.nl/v2/collections/search | jq
```

Output of the example:

```
[  /* Example output has been shortened. */
  {
    "id": 6070238,
    "uuid": "3dfc4ef2-7f79-4d33-81a7-9c6ae09a2782",
    "title": "Flared Folding Wingtips - TU Delft",
    "doi": "10.4121/c.6070238.v1",
    "handle": "",
    "url": "https://data.4tu.nl/v2/collections/3dfc...2782",
    "timeline": {
      "posted": "2023-04-05T15:05:04",
      "firstOnline": "2023-04-05T15:05:03",
      ...
    },
    "published_date": "2023-04-05T15:05:04"
  },
  ...
]
```

5.1.11 /v2/collections/<collection-id> (HTTP GET)

This API endpoint can be used to retrieve detailed metadata for the collection identified by `collection-id`.

Example usage:

```
curl https://data.4tu.nl/v2/collections/3dfc4ef2-7f79-4d33-81a7-9c6ae09a2782 | jq
```

Output of the example:

```
{ /* Example output has been shortened. */
  "version": 3,
  ...
  "description": "<p>This collection contains the results of the work ...",
  "categories": [ ... ],
  "references": [],
  "tags": [ ... ],
  "created_date": "2024-08-08T15:48:55",
  "modified_date": "2024-08-12T11:24:39",
  "id": 6070238,
  "uuid": "3dfc4ef2-7f79-4d33-81a7-9c6ae09a2782",
  "title": "Flared Folding Wingtips - TU Delft",
  "doi": "10.4121/c.6070238.v3",
  "published_date": "2024-08-12T11:24:40",
  "timeline": ...
  ...
}
```

5.1.12 /v2/collections/<collection-id>/versions (HTTP GET)

This API endpoint can be used to retrieve a list of versions for the collection identified by `collection-id`.

Example usage:

```
curl https://data.4tu.nl/v2/collections/3dfc4ef2-7f79-4d33-81a7-9c6ae09a2782/
   versions | jq
```

Output of the example:

```
[ /* Example output has been shortened. */
  {
    "version": 3,
    "url": "https://data.4tu.nl/v2/collections/3dfc...2782/versions/3"
  },
  {
    "version": 2,
    "url": "https://data.4tu.nl/v2/collections/3dfc...2782/versions/2"
  },
  {
    "version": 1,
    "url": "https://data.4tu.nl/v2/collections/3dfc...2782/versions/1"
  }
]
```

5.1.13 /v2/collections/<collection-id>/versions/<version> (HTTP GET)

This API endpoint can be used to retrieve detailed metadata of the version `version` for the collection identified by `collection-id`.

Example usage:

```
curl https://data.4tu.nl/v2/collections/3dfc...2782/versions/2 | jq
```

Output of the example:

```
{ /* Example output has been shortened. */
  "version": 2,
  ...
  "description": "<p>This collection contains the results of the work ...",
  "categories": [ ... ],
  "references": [],
  "tags": [ ... ],
  "references": [],
  "tags": [ ... ],
  "authors": [ ... ],
  "created_date": "2023-04-05T15:07:35",
  "modified_date": "2023-05-26T15:19:11",
  "id": 6070238,
  "uuid": "3dfc4ef2-7f79-4d33-81a7-9c6ae09a2782",
  "title": "Flared Folding Wingtips - TU Delft",
  "doi": "10.4121/c.6070238.v2",
  ...
}
```

5.1.14 /v2/collections/<collection-id>/articles (HTTP GET)

This API endpoint can be used to retrieve the list of datasets in the collection identified by `collection-id`.

Example usage:

```
curl https://data.4tu.nl/v2/collections/3dfc...2782/articles | jq
```

Output of the example:

```
[ /* Example output has been shortened. */
  {
    "id": 20222334,
    "uuid": "c5fde4a2-798a-456e-b793-cf64e486c0e8",
    "title": "E001 - Stiffness and Hinge Release Study (October 2021) ...",
    "doi": "10.4121/20222334.v2",
    "published_date": "2023-05-31T08:57:54",
```

```
    "defined_type": 3,
    "defined_type_name": "dataset",
    "group_id": 28586,
    "timeline": {
      "posted": "2023-05-31T08:57:54",
      "firstOnline": "2023-05-26T15:08:09",
      "revision": null
    },
    "resource_title": "Effect of Wing Stiffness and Folding Wingtip ...",
    "resource_doi": "https://doi.org/10.2514/1.C037108"
  },
  {
    "id": null,
    "uuid": "984090ea-26fd-4809-8dac-f41367bf8916",
    "title": "M001 - GVT Data and Nastran models (August 2024) ...",
    "doi": "10.4121/984090ea-26fd-4809-8dac-f41367bf8916.v1",
    "published_date": "2024-08-12T11:21:47",
    "defined_type": 3,
    "defined_type_name": "dataset",
    "group_id": 28586,
    "timeline": {
      "posted": "2024-08-12T11:21:47",
      "firstOnline": "2024-08-12T11:21:46",
      "revision": null
    },
    "resource_title": "Effect of Wing Stiffness and Folding Wingtip ...",
    "resource_doi": "https://doi.org/10.2514/1.C037108"
  }
]
```


5.1.15 /v2/categories (HTTP GET)

Each dataset and collection is categorized using a controlled vocabulary of categories. This API endpoint provides those categories.

Example usage:

```
curl https://data.4tu.nl/v2/categories | jq
```

Output of the example:


```
[ /* Example output has been shortened. */
  {
    "id": 13622,
    "uuid": "01fddd41-68d2-4e28-9d9c-18347847e7d1",
    "title": "Mining and Extraction of Energy Resources",
```

```
    "parent_id": 13620,
    "parent_uuid": "6e5bdc69-96db-41e4-ac0b-18812b46c49c",
    "path": "",
    "source_id": null,
    "taxonomy_id": null
  },
  {
    "id": 13443,
    "uuid": "026f555c-2826-4a83-97ff-0f230fb54ddb",
    "title": "Livestock Raising",
    "parent_id": 13440,
    "parent_uuid": "45a8c849-ab59-4302-af79-09b8c0677df8",
    "path": "",
    "source_id": null,
    "taxonomy_id": null
  },
  ...
]
```

5.1.16 /v2/licenses (HTTP GET)

Publishing a dataset involves communicating under which conditions it can be re-used. The licenses under which you can publish a dataset can be found with this API endpoint.

Example usage:

```
curl https://data.4tu.nl/v2/licenses | jq
```

Output of the example:

```
[ /* Example output has been shortened. */
  {
    "value": 1,
    "name": "CC BY 4.0",
    "url": "https://creativecommons.org/licenses/by/4.0/",
    "type": "data"
  },
  {
    "value": 10,
    "name": "CC BY-NC 4.0",
    "url": "https://creativecommons.org/licenses/by-nc/4.0/",
    "type": "data"
  },
  ...
]
```

5.2 The /v2 private interface

The interaction with the v2 private interface API requires an API token. Such a token can be obtained from the dashboard page after logging in. This token can then be passed along in the `Authorization` HTTP header as:

```
Authorization: token YOUR_TOKEN_HERE
```

5.2.1 /v2/account/articles (HTTP GET)

This API endpoint lists the draft datasets of the account to which the authorization token belongs.

The following parameters can be used:

Parameter	Required	Description
page	Optional	The page number used in combination with page_size.
page_size	Optional	The number of datasets per page. Used in combination with page.
limit	Optional	The maximum number of datasets to output. Used together with offset.
offset	Optional	The number of datasets to skip in the output. Used together with limit.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles | jq
```

Output of the example:

```
{ /* Example output has been shortened. */
  "id": null,
  "uuid": "6ddd7a31-8ad8-4c20-95a3-e68fe716fa42",
  "title": "Example draft dataset",
  "doi": null,
  "handle": null,
  "url": "https://data.4tu.nl/v2/articles/6ddd7a31-8ad8-4c20-95a3-e68fe716fa42",
  "published_date": null,
  ...
}
```

5.2.2 /v2/account/articles (HTTP POST)

This API endpoint can be used to create a new dataset.

The following parameters can be used:

Parameter	Data type	Description
`title`	`string`	The title of the dataset.
`description`	`string`	A description of the dataset.
`tags`	list of `strings`	Keywords to enhance the findability of the dataset. Instead of using the key `tags`, you may also use the key `keywords`.
`keywords`	list of `strings`	See `tags`.
`references`	list of `strings`	URLs to resources referring to this dataset, or resources that this dataset refers to.
`categories`	list of `strings`	Categories are a controlled vocabulary and can be used to make the dataset findable in the categorical overviews. The `string` values expected here can be found under the `uuid` property with a call to `/v2/categories`. For more details, see section 5.1.15 '`/v2/categories` (HTTP `GET`)'.
`authors`	list of author records	
`defined_type`	`string`	One of: `figure`, `online resource`, `preprint`, `book`, `conference contribution`, `media`, `dataset`, `poster`, `journal contribution`, `presentation`, `thesis` or `software`.
`funding`	`string`	One-liner to cite funding.
`funding_list`	list of funding records	
`license`	`integer`	Licences communicate under which conditions the dataset can be re-used. The `integer` value to submit here can be found as the `value` property in a call to `/v2/licences`. For more details, see section 5.1.16 '`/v2/licenses` (HTTP `GET`)'.
`language`	`string`	An ISO 639-1 language code.
`doi`	`string`	Do not use this field as a DOI will be automatically assigned upon publication..
`handle`	`string`	Do not use this field as it is deprecated.
`resource_doi`	`string`	The URL of the DOI of an associated peer-reviewed journal publication.
`resource_title`	`string`	The title of the associated peer-reviewed journal publication.
`publisher`	`string`	The name of the data repository publishing the dataset.
`custom_fields`	list of key-value pairs	
`timeline`		Do not use this field because it will be automatically populated during the publication process.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "title": "Example dataset" }' \
     https://data.4tu.nl/v2/account/articles | jq
```

Output of the example:

```
{ /* The UUID in this example has been shortened. */
  "location": "https://data.4tu.nl/v2/account/articles/d7b3...995b1",
  "warnings": []
}
```

5.2.3 /v2/account/articles/<dataset-id> (HTTP GET)

This API endpoint lists details of the dataset identified by `dataset-id`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3daa5-45e2-47b0-9910-0f7fa6a995b1
    | jq
```

Output of the example:

```
{ /* Example output has been shortened. */
  "files": [],
  "authors": [],
  "id": null,
  "uuid": "637e9a3b-3e6d-4810-bc8d-f15ab1d6a4d7",
  "title": "Example dataset",
  ...
}
```

5.2.4 /v2/account/articles/<dataset-id> (HTTP PUT)

This API endpoint can be used to update the metadata of the dataset identified by `dataset-id`.

The following parameters can be used:

Parameter	Required	Description
`title`	`string`	The title of the dataset.
`description`	`string`	A description of the dataset.
`resource_doi`	`string`	The URL of the DOI of an associated peer-reviewed journal publication.
`resource_title`	`string`	The title of the associated peer-reviewed journal publication.
`license`	`integer`	Licences communicate under which conditions the dataset can be re-used. The `integer` value to submit here can be found as the `value` property in a call to `/v2/licences`. For more details, see section 5.1.16 '`/v2/licenses` (HTTP GET)'.
`group_id`	`integer`	
`time_coverage`	`string`	Free-text field to describe the time coverage of the dataset.
`publisher`	`string`	The name of the data repository publishing the dataset.
`language`	`string`	An ISO 639-1 language code.
`contributors`	`string`	Free-text field to indicate contributors to the dataset other than direct authors.
`license_remarks`	`string`	Free-text field to clarify licensing details.
`geolocation`	`string`	Free-text field to specify a location.
`longitude`	`string`	The longitude coordinate of the location.
`latitude`	`string`	The latitude coordinate of the location.
`format`	`string`	Free-text field to indicate the data format(s) used in the dataset.
`data_link`	`string`	URL to where the data can be found. This is only applicable when data is not directly uploaded.
`derived_from`	`string`	DOI or URL of a dataset from which this dataset is derived from.
`same_as`	`string`	DOI or URL of the dataset that is the same as this one.
`organizations`	`string`	Free-text field to specify organizations that contributed or are associated with the dataset.
`is_embargoed`	`boolean`	Set to `true` when the dataset is under embargo.
`embargo_options`	`Object`	An `Object` with an `id` property that can have either the integer value `1000` to indicate the dataset has no end-date for the embargo or the integer value `1001` to indicate that the dataset is permanently closed-access.
`embargo_until_date`	`string`	A date indicator for when the dataset will be available publically.
`embargo_type`	`string`	Either `file` for files-only embargo or `article` to also hide the metadata, except for the title and authors of the dataset.
`embargo_title`	`string`	Title of the embargo.

Example usage:

```
curl --verbose --request PUT \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --data '{ "title": "Updated title" }'
     https://data.4tu.nl/v2/account/articles/d7b3daa5-45e2-47b0-9910-0f7fa6a995b1
    | jq
```

HTTP response of the example:

```
HTTP/1.1 205 RESET CONTENT
```

5.2.5 /v2/account/articles/<dataset-id> (HTTP DELETE)

This API endpoint can be used to delete a draft dataset.

Example usage:

```
curl --verbose --request DELETE \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3daa5-45e2-47b0-9910-0f7fa6a995b1
```

HTTP response of the example:

```
HTTP/1.1 204 NO CONTENT
```

5.2.6 /v2/account/articles/<dataset-id>/authors (HTTP GET)

This API endpoint lists the authors of the dataset identified by `dataset-id`. The following URL parameters can be used:

Parameter	Required	Description
`order`	Optional	Field to use for sorting.
`order_direction`	Optional	Can be either `asc` or `desc`.
`limit`	Optional	The maximum number of datasets to output. Used together with `offset`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3daa5-45e2-47b0-9910-0f7fa6a995b1
    | jq
```

Output of the example:

```
[
  {
    "id": null,
    "uuid": "08f4d496-67b5-4b7c-b2d2-923458d1f450",
    "full_name": "John Doe Jr",
    "is_active": false,
    "url_name": null,
    "orcid_id": ""
  },
  {
    "id": null,
    "uuid": "6815031c-21dc-4873-93c9-f6539da482ce",
    "full_name": "John Doe",
    "is_active": false,
    "url_name": null,
    "orcid_id": ""
  }
]
```


5.2.7 /v2/account/articles/<dataset-id>/authors (HTTP POST)

This API endpoint can be used to append authors to the dataset identified by `dataset-id`.

Example usage:

```
curl --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "authors": [{ "name": "John Doe" }]}' \
     https://data.4tu.nl/v2/account/articles/d7b3daa5-45e2-47b0-9910-0f7fa6a995b1
   /authors
curl --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "authors": [{ "name": "John Doe Jr" }]}' \
     https://data.4tu.nl/v2/account/articles/d7b3daa5-45e2-47b0-9910-0f7fa6a995b1
   /authors
```

The following is an example of the output of the HTTP POST calls:

```
HTTP/1.1 205 RESET CONTENT
```

An example of the output of the HTTP GET call can be found in 5.2.6 '/v2/account/articles/<dataset-id>/auth (HTTP GET)'.

5.2.8 /v2/account/articles/<dataset-id>/authors (HTTP PUT)

In contrast to 5.2.7 '/v2/account/articles/<dataset-id>/authors (HTTP POST)', this API endpoint can be used to **overwrite** the list of authors of the dataset identified by `dataset-id`.

Example usage:

```
curl --request PUT \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "authors": [{ "name": "John Doe" }]}' \
     https://data.4tu.nl/v2/account/articles/d7b3daa5-45e2-47b0-9910-0f7fa6a995b1
   /authors

curl --request PUT \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "authors": [{ "name": "John Doe Jr" }]}' \
     https://data.4tu.nl/v2/account/articles/d7b3daa5-45e2-47b0-9910-0f7fa6a995b1
   /authors

curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3daa5-45e2-47b0-9910-0f7fa6a995b1
     | jq
```

Output of the example:

```
[
  {
    "id": null,
    "uuid": "61751fe3-53a1-477f-a46f-e534cbd0b618",
    "full_name": "John Doe Jr",
    "is_active": false,
    "url_name": null,
    "orcid_id": ""
  },
]
```

5.2.9 /v2/account/articles/<dataset-id>/authors/<author-id> (HTTP DELETE)

This API endpoint can be used to delete an author's association with a dataset.

Example usage:

```
curl --request DELETE \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/authors/6175...0b618
```

HTTP response of the example:

```
HTTP/1.1 204 NO CONTENT
```

5.2.10 /v2/account/articles/<dataset-id>/funding (HTTP GET)

This API endpoint lists the funding of the dataset identified by `dataset-id`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3...95b1/funding | jq
```

Output of the example:

```
[
  {
    "id": null,
    "uuid": "6f605fe1-e87a-43f5-8b67-70ebe3f9b868",
    "title": "Example cases fund",
    "grant_code": "EXA-001",
    "funder_name": "Example",
    "is_user_defined": null,
    "url": "https://example.exa"
  }
]
```

5.2.11 /v2/account/articles/<dataset-id>/funding (HTTP POST)

This API endpoint can be used to append funders to the dataset identified by `dataset-id`.

Example usage:

```
curl --verbose --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "funders": [{ "title": "Example cases fund", \
                             "grant_code": "EXA-001", \
                             "funder_name": "Example", \
                             "url": "https://example.exa" }]}' \
     https://data.4tu.nl/v2/account/articles/d7b3daa5-45e2-47b0-9910-0f7fa6a995b1
   /funding
```

HTTP response of the example:

```
HTTP/1.1 205 RESET CONTENT
```

5.2.12 /v2/account/articles/<dataset-id>/funding (HTTP PUT)

In contrast to 5.2.11 '/v2/account/articles/<dataset-id>/funding (HTTP POST)', this API endpoint can be used to **overwrite** the list of funders of the dataset identified by `dataset-id`.

```
curl --verbose --request PUT \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "funders": [{ "title": "Example cases fund",
                             "grant_code": "EXA-001",
                             "funder_name": "Example",
                             "url": "https://example.exa" }]}' \
     https://data.4tu.nl/v2/account/articles/d7b3daa5-45e2-47b0-9910-0f7fa6a995b1
   /funding
```

HTTP response of the example:

```
HTTP/1.1 205 RESET CONTENT
```

5.2.13 /v2/account/articles/<dataset-id>/funding/<funding-id> (HTTP DELETE)

This API endpoint can be used to delete an funder's association with a dataset.

Example usage:

```
curl --request DELETE \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/funding/d50e...7500
```

HTTP response of the example:

```
HTTP/1.1 204 NO CONTENT
```

5.2.14 /v2/account/articles/<dataset-id>/categories (HTTP GET)

This API endpoint lists the categories of the dataset identified by `dataset-id`. The identifiers for the categories can be found by using the API endpoint described at 5.1.15 '/v2/categories (HTTP GET)'.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3...95b1/categories | jq
```

Output of the example:

```
[
  {
```

```
    "id": 13558,
    "uuid": "8f27eb44-0a63-4496-ba6d-e3cbf4efa6c7",
    "title": "Other Earth Sciences",
    "parent_id": 13551,
    "parent_uuid": "dd4dbaaf-0610-4d8d-8b07-e1eeb32dd11c",
    "path": "",
    "source_id": null,
    "taxonomy_id": null
  },
  {
    "id": 13551,
    "uuid": "dd4dbaaf-0610-4d8d-8b07-e1eeb32dd11c",
    "title": "Earth Sciences",
    "parent_id": null,
    "parent_uuid": null,
    "path": "",
    "source_id": null,
    "taxonomy_id": null
  }
]
```

5.2.15 /v2/account/articles/<dataset-id>/categories (HTTP POST)

This API endpoint can be used to append categories to the dataset identified by `dataset-id`.

Example usage:

```
curl --verbose --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "categories": [13551, 13558]}' \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/categories
```

HTTP response of the example:

```
HTTP/1.1 205 RESET CONTENT
```

5.2.16 /v2/account/articles/<dataset-id>/categories (HTTP PUT)

In contrast to 5.2.15 '/v2/account/articles/<dataset-id>/categories (HTTP POST)', this API endpoint can be used to **overwrite** the list of categories of the dataset identified by `dataset-id`.

Example usage:

```
curl --verbose --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
```

```
     --header "Content-Type: application/json" \
     --data '{ "categories": ["dd4dbaaf-0610-4d8d-8b07-e1eeb32dd11c"]}' \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/categories
```

HTTP response of the example:

```
HTTP/1.1 205 RESET CONTENT
```

5.2.17 /v2/account/articles/<dataset-id>/categories/<category-id> (HTTP DELETE)

This API endpoint can be used to delete a category's association with a dataset. The `category-id` can be either the `uuid` or the `id` property.

Example usage:

```
curl --request DELETE \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/categories/5c61...b668
```

HTTP response of the example:

```
HTTP/1.1 204 NO CONTENT
```

5.2.18 /v2/account/articles/<dataset-id>/embargo (HTTP GET)

This API endpoint lists the embargo status of the dataset identified by `dataset-id`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/embargo | jq
```

Output of the example:

```
{
  "is_embargoed": false,
  "embargo_date": null,
  "embargo_type": "file",
  "embargo_title": "",
  "embargo_reason": "",
  "embargo_options": []
}
```

5.2.19 /v2/account/articles/<dataset-id>/embargo (HTTP DELETE)

This API endpoint can be used to remove an embargo on the dataset identified by `dataset-id`.

Example usage:

```
curl --verbose --request DELETE \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3d...995b1/embargo
```

HTTP response of the example:

```
HTTP/1.1 204 NO CONTENT
```

5.2.20 /v2/account/articles/<dataset-id>/files (HTTP GET)

This API endpoint lists files associated with the dataset identified by `dataset-id`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/files | jq
```

Output of the example:

```
[ /* Example output has been shortened. */
  {
    "id": null,
    "uuid": "d112d0cd-bc15-4f8e-9013-930750fc017a",
    "name": "README.md",
    "size": 3696,
    "is_link_only": false,
    "is_incomplete": false,
    "download_url": "https://next.data.4tu.nl/file/d7b3...995b1/d112...c017a",
    "supplied_md5": null,
    "computed_md5": "c5b36584a0d62d28e9bf9e6892d9ebac"
  }
]
```

5.2.21 /v2/account/articles/<dataset-id>/files (HTTP DELETE)

Note: This API endpoint is a djehuty extension to the original specification.

This API endpoint can be used to delete all files associated with the dataset identified by `dataset-id`.

Example usage:

```
curl --request DELETE \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "remove_all": true }' \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/files
```

HTTP response of the example:

```
HTTP/1.1 204 NO CONTENT
```

5.2.22 /v2/account/articles/<dataset-id>/files/<file-id> (HTTP GET)

This API endpoint lists files associated with the dataset identified by `dataset-id`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/files | jq
```

Output of the example:

```
[ /* Example output has been shortened. */
  {
    "id": null,
    "uuid": "d112d0cd-bc15-4f8e-9013-930750fc017a",
    "name": "README.md",
    "size": 3696,
    "is_link_only": false,
    "is_incomplete": false,
    "download_url": "https://next.data.4tu.nl/file/d7b3...995b1/d112...c017a",
    "supplied_md5": null,
    "computed_md5": "c5b36584a0d62d28e9bf9e6892d9ebac"
  }
]
```

5.2.23 /v2/account/articles/<dataset-id>/private_links (HTTP GET)

This API endpoint lists the private links associated with the dataset identified by `dataset-id`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/private_links | jq
```

Output of the example:

```
[
  {
    "id": "8G2f...IJP0",
    "is_active": true,
    "expires_date": "2032-01-01T00:00:00"
  },
  {
    "id": "Hb0a...diitg",
    "is_active": true,
    "expires_date": "2026-01-01T00:00:00"
  }
]
```

5.2.24 /v2/account/articles/<dataset-id>/private_links (HTTP POST)

This API endpoint can be used to append a private link to the dataset identified by `dataset-id` .

Example usage:

```
curl --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --data '{ "expires_date": "2032-01-01", "read_only": false }' \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/private_links | jq
```

output of the example:

```
{ /* Example output has been shortened. */
  "location": "https://data.4tu.nl/private_datasets/8G2fk..."
}
```

5.2.25 /v2/account/articles/<dataset-id>/private_links/<link-id> (HTTP GET)

This API endpoint can be used to view the details of a private link for the dataset identified by `dataset-id` .

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/private_links/8G2fk...
   | jq
```

Output of the example:

```
[
  {
    "id": "8G2f...IJP0",
    "is_active": true,
    "expires_date": "2032-01-01T00:00:00"
  }
]
```

5.2.26 /v2/account/articles/<dataset-id>/private_links/<link-id> (HTTP PUT)

This API endpoint can be used to update the expiry date of a private link and whether the private link is active or not for the dataset identified by `dataset-id`.

Example usage:

```
curl --request PUT \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --data '{ "expires_date": "2034-01-01", "is_active": true }' \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/private_links/8G2fk...
   | jq
```

Output of the example:

```
{ /* Example output has been shortened. */
  "location": "https://data.4tu.nl/private_datasets/8G2fk..."
}
```

5.2.27 /v2/account/articles/<dataset-id>/private_links/<link-id> (HTTP DELETE)

This API endpoint can be used to remove a private link for the dataset identified by `dataset-id`.

Example usage:

```
curl --request DELETE \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/private_links/8G2fk...
```

HTTP response of the example:

```
HTTP/1.1 204 NO CONTENT
```

5.2.28 /v2/account/articles/<dataset-id>/reserve_doi (HTTP POST)

This API endpoint can be used to obtain the DOI before the dataset is published and the DOI is activated.

Example usage:

```
curl --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/reserve_doi | jq
```

Output of the example:

```
{
  "doi": "10.5074/d7b3daa5-45e2-47b0-9910-0f7fa6a995b1"
}
```

5.2.29 /v2/account/articles/<dataset-id>/publish (HTTP POST)

This API endpoint can be used to publish the dataset identified by `dataset-id`.

Note: Only users with the "review" privilege can succesfully use this API call.

Example usage:

```
curl --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/articles/d7b3...995b1/publish | jq
```

HTTP response of the example:

```
HTTP/1.1 201 CREATED
```

Output of the example:

```
{ /* Example output has been shortened. */
  "location": "https://data.4tu.nl/review/published/9ce6...3976"
}
```

5.2.30 /v2/account/authors/search (HTTP POST)

This API endpoint can be used to search for authors known to the data repository.

Example usage:

```
curl --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --data '{ "search": "John Doe" }' \
```

```
    https://data.4tu.nl/v2/account/authors/search | jq
```

Output of the example:

```
[ /* This example output has been shortened. */
  {
    "full_name": "John Doe Jr",
    "uuid": "08f4d496-67b5-4b7c-b2d2-923458d1f450",
    "orcid_id": "",
    ...
  },
  {
    "full_name": "John Doe",
    "uuid": "6815031c-21dc-4873-93c9-f6539da482ce",
    "orcid_id": "",
    ...
  }
]
```

5.2.31 /v2/account/authors/<author-id> (HTTP GET)

This API endpoint returns a detailed author record for the author identified by `author-id`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
    https://data.4tu.nl/v2/account/authors/5c75...94aa | jq
```

Output of the example:

```
{ /* This example output has been shortened. */
  "first_name": "Roel",
  "full_name": "Roel Janssen",
  "uuid": "5c752155-60ff-41d7-9b88-b7112afc94aa",
  "last_name": "Janssen",
  "orcid_id": "0000-0003-4324-5350",
  ...
}
```

5.2.32 /v2/account/collections (HTTP GET)

This API endpoint lists the draft collections of the account to which the authorization token belongs.

The following parameters can be used:

Parameter	Required	Description
page	Optional	The page number used in combination with page_size.
page_size	Optional	The number of datasets per page. Used in combination with page.
limit	Optional	The maximum number of datasets to output. Used together with offset.
offset	Optional	The number of datasets to skip in the output. Used together with limit.
order	Optional	Field to use for sorting.
order_direction	Optional	Can be either `asc` or `desc`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/collections | jq
```

Output of the example:

```
[ /* This example output has been shortened. */
  {
    "id": null,
    "uuid": "fc03a4c3-cba4-4a88-a8a6-eb38924eeb6d",
    "title": "Test collection",
    "doi": null,
    "handle": "",
    "url": "https://data.4tu.nl/v2/collections/fc03...eb6d",
    "published_date": null,
    ...
  }
]
```

5.2.33 /v2/account/collections (HTTP POST)

This API endpoint can be used to create a new collection.

The following parameters can be used:

Parameter	Data type	Description
`title`	`string`	The title of the collection.
`description`	`string`	A description of the collection.
`tags`	list of `strings`	Keywords to enhance the findability of the collection. Instead of using the key `tags`, you may also use the key `keywords`.
`references`	list of `strings`	URLs to resources referring to this collection, or resources that this collection refers to.
`categories`	list of `strings`	Categories are a controlled vocabulary and can be used to make the collection findable in the categorical overviews. The `string` values expected here can be found under the `uuid` property with a call to `/v2/categories`. For more details, see section 5.1.15 '`/v2/categories` (HTTP GET)'.
`authors`	list of author records	
`funding`	`string`	One-liner to cite funding.
`funding_list`	list of funding records	
`license`	`integer`	Licences communicate under which conditions the collection can be re-used. The `integer` value to submit here can be found as the `value` property in a call to `/v2/licences`. For more details, see section 5.1.16 '`/v2/licenses` (HTTP GET)'.
`doi`	`string`	Do not use this field as a DOI will be automatically assigned upon publication..
`handle`	`string`	Do not use this field as it is deprecated.
`resource_doi`	`string`	The URL of the DOI of an associated peer-reviewed journal publication.
`resource_title`	`string`	The title of the associated peer-reviewed journal publication.
`custom_fields`	`Object`	An `Object` where each key is a field name and each value is the corresponding value. Allowed values are: `contributors`, `data_link`, `derived_from`, `format`, `geolocation`, `language`, `latitude`, `longitude`, `organizations`, `publisher`, `same_as`, `time_coverage`.
`custom_fields_list`	list of `Objects`	Each `Object` should have two keys: `name` and `value`. For allowed key values, see `custom_fields`.
`timeline`		Do not use this field because it will be automatically populated during the publication process.
`articles`	list of strings or integers	The articles to include in the collection.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "title": "Example collection" }' \
     https://data.4tu.nl/v2/account/collections | jq
```

Output of the example:

```
{
  "location": "https://data.4tu.nl/v2/account/collections/08b7...cfa8",
  "warnings": []
}
```

5.2.34 /v2/account/collections/<collection-id> (HTTP GET)

This API endpoint lists details of the collection identified by `collection-id`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/collections/08b7...cfa8 | jq
```

Output of the example:

```
{ /* Example output has been shortened. */
  "articles_count": 0,
  "authors": [],
  "id": null,
  "uuid": "08b702d6-98a0-4081-9445-5aeae720cfa8",
  "title": "Example collection",
  ...
}
```

5.2.35 /v2/account/collections/<collection-id> (HTTP PUT)

This API endpoint can be used to update the metadata of the collection identified by `collection-id`.

The following parameters can be used:

Parameter	Data type	Required	Description
`title`	`string`	Yes	The title of the collection.
`description`	`string`	No	A description of the collection.
`resource_doi`	`string`	No	The URL of the DOI of an associated peer-reviewed journal publication.
`resource_title`	`string`	No	The title of the associated peer-reviewed journal publication.
`group_id`	`integer`	No	
`time_coverage`	`string`	No	Free-text field to describe the time coverage of the collection.
`publisher`	`string`	No	The name of the data repository publishing the collection.
`language`	`string`	No	An ISO 639-1 language code.
`contributors`	`string`	No	Free-text field to indicate contributors to the collection other than direct authors.
`geolocation`	`string`	No	Free-text field to specify a location.
`longitude`	`string`	No	The longitude coordinate of the location.
`latitude`	`string`	No	The latitude coordinate of the location.
`organizations`	`string`	No	Free-text field to specify organizations that contributed or are associated with the collection.
`categories`	list of `strings`	No	Categories are a controlled vocabulary and can be used to make the collection findable in the categorical overviews. The `string` values expected here can be found under the `uuid` property with a call to `/v2/categories`. For more details, see section 5.1.15 '`/v2/categories` (HTTP GET)'.

Example usage:

```
curl --verbose --request PUT \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "title": "Updated title" }' \
     https://data.4tu.nl/v2/account/collections/08b702d6-98a0-4081-9445-5
   aeae720cfa8 | jq
```

HTTP response of the example:

```
HTTP/1.1 205 RESET CONTENT
```

5.2.36 /v2/account/collections/<collection-id> (HTTP DELETE)

This API endpoint can be used to delete a draft collection.

Example usage:

```
curl --verbose --request DELETE \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/collections/08b702d6-98a0-4081-9445-5
   aeae720cfa8
```

HTTP response of the example:

```
HTTP/1.1 204 NO CONTENT
```

5.2.37 /v2/account/collections/search (HTTP POST)

This API call searches for collections, including drafts created by the account performing the search.

Example usage:

```
curl --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "search_for": "Example" }' \
     https://data.4tu.nl/v2/account/collections/search | jq
```

Output of the example:

```
[ /* Example output has been shortened. */
  {
    "id": null,
    "uuid": "08b702d6-98a0-4081-9445-5aeae720cfa8",
    "title": "Example collection",
    "url": https://data.4tu.nl/v2/collections/08b7...cfa8
    ...
  }
]
```

5.2.38 /v2/account/collections/<collection-id>/authors (HTTP GET)

Similar to 5.2.6 '/v2/account/articles/<dataset-id>/authors (HTTP GET)', this API endpoint lists the authors of the collection identified by `collection-id`. The following URL parameters can be used:

Parameter	Required	Description
`order`	Optional	Field to use for sorting.
`order_direction`	Optional	Can be either `asc` or `desc`.
`limit`	Optional	The maximum number of datasets to output. Used together with `offset`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/collections/3760c457-d4f3-4d58-8b94-
   af089a97a9b4 | jq
```

Output of the example:

```
[
  {
    "id": null,
    "uuid": "08f4d496-67b5-4b7c-b2d2-923458d1f450",
    "full_name": "John Doe Jr",
    "is_active": false,
    "url_name": null,
    "orcid_id": ""
  },
  {
    "id": null,
    "uuid": "6815031c-21dc-4873-93c9-f6539da482ce",
    "full_name": "John Doe",
    "is_active": false,
    "url_name": null,
    "orcid_id": ""
  }
]
```

5.2.39 `/v2/account/collections/<collection-id>/authors` (HTTP POST)

Similar to 5.2.7 '`/v2/account/articles/<dataset-id>/authors` (HTTP POST)', this API endpoint can be used to append authors to the collection identified by `collection-id`.

Example usage:

```
curl --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "authors": [{ "name": "John Doe" }]}' \
```

```
     https://data.4tu.nl/v2/account/collections/3760c457-d4f3-4d58-8b94-
   af089a97a9b4/authors
curl --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "authors": [{ "name": "John Doe Jr" }]}' \
     https://data.4tu.nl/v2/account/collections/3760c457-d4f3-4d58-8b94-
   af089a97a9b4/authors
```

The following is an example of the output of the HTTP POST calls:

```
HTTP/1.1 205 RESET CONTENT
```

An example of the output of the HTTP GET call can be found in 5.2.38 '/v2/account/collections/<collection-i (HTTP GET)'.

5.2.40 /v2/account/collections/<collection-id>/authors (HTTP PUT)

In contrast to 5.2.39 '/v2/account/collections/<collection-id>/authors (HTTP POST)', this API endpoint can be used to **overwrite** the list of authors of the collection identified by `collection-id`.

Example usage:

```
curl --request PUT \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "authors": [{ "name": "John Doe" }]}' \
     https://data.4tu.nl/v2/account/collections/3760c457-d4f3-4d58-8b94-
   af089a97a9b4/authors

curl --request PUT \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "authors": [{ "name": "John Doe Jr" }]}' \
     https://data.4tu.nl/v2/account/collections/3760c457-d4f3-4d58-8b94-
   af089a97a9b4/authors

curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/collections/3760c457-d4f3-4d58-8b94-
   af089a97a9b4/authors | jq
```

Output of the example:

```
[
  {
    "id": null,
```

```
    "uuid": "61751fe3-53a1-477f-a46f-e534cbd0b618",
    "full_name": "John Doe Jr",
    "is_active": false,
    "url_name": null,
    "orcid_id": ""
  },
]
```

5.2.41 /v2/account/collections/<collection-id>/authors/<author-id> (HTTP DELETE)

This API endpoint can be used to delete an author's association with a collection.

Example usage:

```
curl --request DELETE \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/collections/fc03...eb6d//authors/5c75...94aa
```

HTTP response of the example:

```
HTTP/1.1 204 NO CONTENT
```

5.2.42 /v2/account/collections/<collection-id>/categories (HTTP GET)

This API endpoint can be used to retrieve the categories associated with the collection identified by `collection-id`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/collections/fc03...eb6d/categories | jq
```

Output of the example:

```
[
  {
    "id": 13376,
    "uuid": "2bdba8f2-5914-4d82-bfe8-c938cccab71f",
    "title": "Agricultural and Veterinary Sciences",
    "parent_id": null,
    "parent_uuid": null,
    "path": "",
    "source_id": null,
    "taxonomy_id": null
```

```
  }
]
```

5.2.43 /v2/account/collections/<collection-id>/categories (HTTP POST)

Similar to 5.2.15 '/v2/account/articles/<dataset-id>/categories (HTTP POST)' this API endpoint can be used to append categories to the collection identified by `collection-id`.

Example usage:

```
curl --verbose --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "categories": [13551, 13558]}' \
     https://data.4tu.nl/v2/account/collections/fc03...eb6d/categories
```

HTTP response of the example:

```
HTTP/1.1 205 RESET CONTENT
```

5.2.44 /v2/account/collections/<collection-id>/categories (HTTP PUT)

In contrast to 5.2.43 '/v2/account/collections/<collection-id>/categories (HTTP POST)', this API endpoint can be used to **overwrite** the list of categories of the collection identified by `collection-id`.

Example usage:

```
curl --verbose --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "categories": ["dd4dbaaf-0610-4d8d-8b07-e1eeb32dd11c"]}' \
     https://data.4tu.nl/v2/account/collections/fc03...eb6d/categories
```

HTTP response of the example:

```
HTTP/1.1 205 RESET CONTENT
```

5.2.45 /v2/account/collections/<collection-id>/categories/<category-id> (HTTP DELETE)

This API endpoint can be used to delete a category's association with a collection. The `category-id` can be either the `uuid` or the `id` property.

Example usage:

```
curl --request DELETE \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/collections/fc03...eb6d/categories/13558
```

HTTP response of the example:

```
HTTP/1.1 204 NO CONTENT
```

5.2.46 /v2/account/collections/<collection-id>/articles (HTTP GET)

This API endpoint can be used to retrieve the datasets associated with the collection identified by `collection-id`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/collections/fc03...eb6d/articles | jq
```

Output of the example:

```
[ /* This example has been shortened. */
  {
    "id": null,
    "uuid": "8050f9cb-d0b0-4149-bd24-02f13c2410db",
    "doi": "10.4121/8050f9cb-d0b0-4149-bd24-02f13c2410db.v1",
    ...
  },
  {
    "id": 14309234,
    "uuid": "06431360-776c-45c6-bcca-ec898f2870ff",
    "doi": "10.4121/14309234.v1",
    ...
  }
]
```

5.2.47 /v2/account/collections/<collection-id>/articles (HTTP POST)

This API endpoint can be used to append datasets to the collection identified by `collection-id`. The API endpoint accepts both the `id` property and the `uuid` property of a dataset as identifier.

```
curl --verbose --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
```

```
    --header "Accept: application/json" \
    --data '{ "articles": ["8050...10db", 14309234 ]}' \
    https://data.4tu.nl/v2/account/collections/fc03...eb6d/articles
```

HTTP response of the example:

```
HTTP/1.1 205 RESET CONTENT
```

5.2.48 /v2/account/collections/<collection-id>/articles (HTTP PUT)

In contrast to 5.2.47 '/v2/account/collections/<collection-id>/articles (HTTP POST)', this API endpoint can be used to **overwrite** the list of datasets associated with a collection.

```
curl --verbose --request PUT \
    --header "Authorization: token YOUR_TOKEN_HERE" \
    --header "Content-Type: application/json" \
    --header "Accept: application/json" \
    --data '{ "articles": [ 14309234 ]}' \
    https://data.4tu.nl/v2/account/collections/fc03...eb6d/articles
```

HTTP response of the example:

```
HTTP/1.1 205 RESET CONTENT
```

5.2.49 /v2/account/collections/<collection-id>/articles/<dataset-id> (HTTP DELETE)

This API endpoint can be used to delete a dataset's association with a collection. The `dataset-id` can be either the `uuid` or the `id` property.

Example usage:

```
curl --request DELETE \
    --header "Authorization: token YOUR_TOKEN_HERE" \
    https://data.4tu.nl/v2/account/collections/fc03...eb6d/articles/8050...10db
```

HTTP response of the example:

```
HTTP/1.1 204 NO CONTENT
```

5.2.50 /v2/account/collections/<collection-id>/reserve_doi (HTTP POST)

This API endpoint can be used to obtain the DOI before the collection is published and the DOI is activated.

Example usage:

```
curl --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/collections/fc03...eb6d/reserve_doi | jq
```

Output of the example:

```
{
  "doi": "10.5074/fc03a4c3-cba4-4a88-a8a6-eb38924eeb6d"
}
```

5.2.51 /v2/account/collections/<collection-id>/funding (HTTP GET)

This API endpoint lists the funding of the collection identified by `collection-id`.

Example usage:

```
curl --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/collections/fc03...eb6d/funding | jq
```

Output of the example:

```
[
  {
    "id": null,
    "uuid": "6f605fe1-e87a-43f5-8b67-70ebe3f9b868",
    "title": "Example cases fund",
    "grant_code": "EXA-001",
    "funder_name": "Example",
    "is_user_defined": null,
    "url": "https://example.exa"
  }
]
```

5.2.52 /v2/account/collections/<collection-id>/funding (HTTP POST)

This API endpoint can be used to append funders to the collection identified by `collection-id`.

Example usage:

```
curl --verbose --request POST \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "funders": [{ "title": "Example cases fund", \
                             "grant_code": "EXA-001", \
```

```
                              "funder_name": "Example", \
                              "url": "https://example.exa" }]}' \
    https://data.4tu.nl/v2/account/collections/fc03a4c3-cba4-4a88-a8a6-
   eb38924eeb6d/funding
```

HTTP response of the example:

```
HTTP/1.1 205 RESET CONTENT
```

5.2.53 /v2/account/collections/<collection-id>/funding (HTTP PUT)

In contrast to 5.2.52 '/v2/account/collections/<collection-id>/funding (HTTP POST)', this API endpoint can be used to **overwrite** the list of funders of the collection identified by `collection-id`.

```
curl --verbose --request PUT \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     --header "Content-Type: application/json" \
     --data '{ "funders": [{ "title": "Example cases fund",
                             "grant_code": "EXA-001",
                             "funder_name": "Example",
                             "url": "https://example.exa" }]}' \
     https://data.4tu.nl/v2/account/collections/fc03a4c3-cba4-4a88-a8a6-
   eb38924eeb6d/funding
```

HTTP response of the example:

```
HTTP/1.1 205 RESET CONTENT
```

5.2.54 /v2/account/collections/<collection-id>/funding/<funding-id> (HTTP DELETE)

This API endpoint can be used to delete an funder's association with a collection.

Example usage:

```
curl --request DELETE \
     --header "Authorization: token YOUR_TOKEN_HERE" \
     https://data.4tu.nl/v2/account/collections/fc03...eb6d/funding/9b43...e6cd
```

HTTP response of the example:

```
HTTP/1.1 204 NO CONTENT
```

References

Lassila, O. (1999, February). Resource description framework (RDF) model and syntax specification [W3C Recommendation]. (http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/)

SPARQL 1.1 overview [W3C Recommendation]. (2013, March). (http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/)

Contact

Stay up-to-date

Subscribe to our news mailing list to receive release updates and other announcements regarding `djehuty`.

General inquiries

For questions about the project or requesting an (test) instance, e-mail us at info@djehuty.4tu.nl.

Reporting security vulnerabilities

For security-related matters, please e-mail us at security@djehuty.4tu.nl. This will reach only the security teams at 4TU.ResearchData and Nikhef.

Meet the team

We would love to have you over at either TU Delft or Nikhef for a cup of coffee, to talk about the endless possibilities of `djehuty` and what it could mean for you!

Catharina Vaendel
cvaendel@nikhef.nl
Software Infrastructure Engineer @ Nikhef
LinkedIn

Roel Janssen
r.r.e.janssen@tudelft.nl
Senior Software Engineer @ TU Delft
Github

News

Release notes for `v25.1`.

The January release of 2025 consists of 85 commits made by 3 authors.

In this release we included an RPM package for Enterprise Linux 9. This RPM depends on packages in the Extra Packages for Enterprise Linux (EPEL) repository.

New features

- CodeMeta API output is more complete (91ed59d86, 860edfec5, 3f150a246, 5daa47e5f).
- A second port to bind the web service on can now be configured (dff68a6d6).
- Enable searching by author (18a242fcd, fe0957865).
- Enable institution reviewing (cbd1092cc, cb67f83bf).
- Improve indexability by search engines (cc5848390, 20fe6fd9a).

Bugfixes

- Related versions of a dataset are communicated to DataCite (1539117be).
- HTML output of the documentation is responsive to browser widths (934ed9310).
- Restore ability to create new collection versions (545de472c).
- Display embargoed datasets in the search results (c86dcf85b).
- Fixed building RPM packages (1f629fa0d, e4c10571a).
- Fixed HTTP `PUT` behavior for `/v2/account/collections/<id>/articles` (107ea693e.)

Technical debt

- Unified the development environment instructions between GNU/Linux, Windows and macOS (8921d35c1, ca9b5831c).
- Run-time configurable properties are stored in a separate module (a8e353db6).
- Improve error handling (bdf77edd8, 23c3b53d2, 609c9864e).
- Embed simplified 'zipfly' (0fe0904a2).

Release notes for v24.12.

New features

- Add specific logging for when the server would respond an HTTP 500 error (1dcd141f7).

Bugfixes

- Fix a problem with downloading Git repositories as ZIP (a8c2da4c5).
- Avoid returning an internal server error when using paging in the API (2629ef56f).
- Fix lay-out bug on the landing pages (a3ff6a54e).
- Fix bug when filtering on groups in the API (187f4344f).

Technical debt

- Refactor parts of the codebase (0370836d5, dd7602a66, aa8eda2fc, 3fb51635c, dd6ccca45, 58c1fcedc)
- Reduce build-system files (e73966fed).
- Bump the minimal required Python version to 3.9 (208c7e08f).