Dataverse Documentation

Release 6.12

Dataverse Team

Sep 30, 2026

CONTENTS

1 Quickstart Guide **3**
- 1.1 What is Dataverse? . . . 3
- 1.2 Publish a Dataset . . . 4
- 1.3 Publish a Collection . . . 5

2 User Guide **9**
- 2.1 Account Creation + Management . . . 9
- 2.2 Finding and Using Data . . . 16
- 2.3 Dataverse Collection Management . . . 23
- 2.4 Dataset + File Management . . . 36
- 2.5 Tabular Data File Ingest . . . 68
- 2.6 Appendix . . . 79

3 Admin Guide **81**
- 3.1 Features . . . 81
- 3.2 Dashboard . . . 86
- 3.3 External Tools . . . 87
- 3.4 Discoverability . . . 93
- 3.5 Managing Harvesting Clients . . . 95
- 3.6 Managing Harvesting Server and Sets . . . 97
- 3.7 Metadata Customization . . . 100
- 3.8 Metadata Export . . . 112
- 3.9 Dataverse Installation Application Timers . . . 114
- 3.10 Make Data Count . . . 116
- 3.11 Integrations . . . 121
- 3.12 User Administration . . . 128
- 3.13 Managing Datasets and Dataverse Collections . . . 132
- 3.14 Solr Search Index . . . 142
- 3.15 IP Groups . . . 145
- 3.16 Mail Domain Groups . . . 147
- 3.17 Storage Quotas for Collections . . . 149
- 3.18 Monitoring . . . 150
- 3.19 Rate Limiting . . . 154
- 3.20 Reporting Tools and Common Queries . . . 156
- 3.21 Maintenance . . . 156
- 3.22 Backups . . . 156
- 3.23 Troubleshooting . . . 157
- 3.24 Scaling Dataverse with Data Size . . . 162

4 AI Guide **173**

4.1 Tools . . . 173
4.2 Protocols . . . 174

5 API Guide **175**
5.1 Introduction . . . 175
5.2 Getting Started with APIs . . . 180
5.3 API Tokens and Authentication . . . 185
5.4 Search API . . . 188
5.5 Data Access API . . . 206
5.6 Native API . . . 218
5.7 Metrics API . . . 408
5.8 SWORD API . . . 419
5.9 Client Libraries . . . 427
5.10 Building External Tools . . . 430
5.11 Dataset Curation Status API . . . 439
5.12 Linked Data Notification API . . . 442
5.13 Apps . . . 444
5.14 Frequently Asked Questions . . . 448
5.15 API Changelog (Breaking Changes) . . . 451

6 Installation Guide **457**
6.1 Introduction . . . 457
6.2 Preparation . . . 459
6.3 Prerequisites . . . 464
6.4 Installation . . . 475
6.5 Configuration . . . 480
6.6 Big Data Support . . . 615
6.7 Upgrading . . . 622
6.8 Shibboleth . . . 622
6.9 OAuth Login Options . . . 636
6.10 OpenID Connect Login Options . . . 640
6.11 ORCID Integration . . . 643
6.12 External Tools . . . 644
6.13 LocalContexts Integration . . . 644
6.14 Advanced Installation . . . 646

7 Contributor Guide **651**
7.1 Ideas and Feature Requests . . . 651
7.2 Bug Reports . . . 652
7.3 Documentation . . . 652
7.4 Translations . . . 657
7.5 Code . . . 657
7.6 Usability Testing . . . 659
7.7 Answering Questions . . . 660
7.8 Sample Data . . . 660
7.9 Issue Triage . . . 660
7.10 Giving Talks . . . 660
7.11 Working Groups . . . 660
7.12 GDCC . . . 660
7.13 Artwork . . . 661
7.14 Other Contributions . . . 661

8 Developer Guide **663**
8.1 Introduction . . . 663
8.2 Development Environment . . . 665

8.3 Windows Development . . . 667
8.4 Tips . . . 670
8.5 Troubleshooting . . . 677
8.6 Version Control . . . 680
8.7 SQL Upgrade Scripts . . . 688
8.8 Testing . . . 689
8.9 API Design . . . 703
8.10 Security . . . 705
8.11 Performance . . . 706
8.12 Dependency Management . . . 712
8.13 Debugging . . . 721
8.14 Coding Style . . . 722
8.15 Consuming Configuration . . . 726
8.16 Deployment . . . 728
8.17 Docker, Kubernetes, and Containers . . . 732
8.18 Making Releases . . . 733
8.19 Making Library Releases . . . 744
8.20 Metadata Export Formats . . . 747
8.21 Tools . . . 749
8.22 Universal Numerical Fingerprint (UNF) . . . 756
8.23 Make Data Count . . . 763
8.24 Shibboleth, OAuth and OIDC . . . 767
8.25 Geospatial Data . . . 768
8.26 SELinux . . . 770
8.27 Auxiliary File Support . . . 774
8.28 Direct DataFile Upload/Replace API . . . 775
8.29 Globus Transfer API . . . 782
8.30 Dataset Semantic Metadata API . . . 788
8.31 Dataset Migration API . . . 790
8.32 Workflows . . . 792
8.33 Font Custom . . . 797
8.34 Classic Dev Environment . . . 799
8.35 Search Services . . . 805
8.36 Fixtures For Tests . . . 808
8.37 Performance Testing . . . 817

9 Container Guide **821**
9.1 Introduction . . . 821
9.2 Running Dataverse in Docker . . . 822
9.3 Development Usage . . . 832
9.4 Base Image . . . 850
9.5 Application Image . . . 856
9.6 Config Baker Image . . . 861

10 Style Guide **867**
10.1 Foundations . . . 867
10.2 Patterns . . . 872
10.3 Text . . . 882

11 How the Guides Are Organized **883**

12 Other Resources **885**

13 Indices and Tables **887**

Bibliography 889

Check out the new *Quickstart Guide*!

These documentation guides are for the 6.12 version of Dataverse. To find guides belonging to previous or future versions, guides_versions has a list of all available versions.

CHAPTER

ONE

QUICKSTART GUIDE

1.1 What is Dataverse?

Dataverse is an open source web application for sharing, preserving, citing, exploring, and analyzing research data. It is developed and supported by the Dataverse user community.

A Dataverse repository can host one or more Dataverse collections, which organize datasets.

- Collections can contain datasets and sub-collections for further organization.
- Each dataset includes:
  - Metadata
  - Data files
  - Documentation or code

1.1.1 Core Capabilities

Upload, manage, publish and download data files.

- Upload data while retaining directory structure for better context and reproducibility.
- Manage datasets by inviting collaborators before publication.
- Control access with permissions, configurations, licenses, file restrictions, and guestbooks.
- Publish datasets with rich metadata, licensing, and versioning to make data FAIR.
- Download data with clear terms of use and cite data using provided citation options.

Make your data findable, reusable, and citable with rich metadata

- Rich metadata is added to a dataset before publication, with the option to use domain-specific metadata blocks.
- Harvest metadata of datasets through the distribution of standardized data descriptions across the web (e.g. Google Dataset Search).

Define how data can be reused with clear terms

- Standardized licenses can be applied to a dataset.
- Custom dataset terms allow for dataset publication that cannot use standardized licenses.

Keep track of changes on published datasets with versioning

- Track version changes with metadata and file changes tracked as minor and major versions.
- Access and cite specific versions via the version tab on a dataset.
- Compare versions with the detailed version change overview on dataset-level.

More features

The Dataverse project is continuously evolving. For an overview of capabilities, see *Features* in the Admin Guide.

1.2 Publish a Dataset

1.2.1 Step 1: Log In & Create a Draft

- *Log in*.
- (Optional) Navigate to *a collection*.
- Click "Add Data" → "*New Dataset*".

Note: If you don't see the "Add Data" button, contact your repository support team.

1.2.2 Step 2: Enter Basic Metadata

- Fill in the required *metadata* fields.
- Click "Save" at the bottom to save your draft dataset.

1.2.3 Step 3: Upload or Edit Files

- In the draft dataset, scroll down to the "Files" tab.
- Click "*Upload Files*".
- Choose "Select Files to Add" or drag and drop files.
- (Optional) Use "*Upload Folder*" if available.
- Click "Done" when upload is completed.

To edit files:

- Select files on the left → Click "*Edit Files*".
- To *restrict/embargo files*, choose the relevant option.

1.2.4 Step 4: Set Terms of Use

- Go to the terms tab.
- Click "*Edit Terms Requirements*".
- *Choose a license* from the dropdown or select *Custom Dataset Terms*.
- Click "Save changes".

1.2.5 Step 5: Add or Edit Metadata

- Go to the metadata tab.
- Click "Add + Edit Metadata".
- Click "Save" after making changes.

1.2.6 Step 6: Publish Your Dataset

Note: once published, easy deletion of a dataset is no longer possible.

- Click publish-dataset or submit-for-review (top right).
- *Publish Dataset*: Immediately publish a dataset. This option is only available on repositories without a review phase.
- *Submit for Review*: Locks draft and sends to support staff. If changes are needed, you'll be notified via email and you can resubmit. This option is only available on repositories with a review phase.

1.2.7 Optional: Update a Published Dataset

- Edit the dataset and publish *a new version*.
- The DOI remains the same.
- The dataset-versioning tracks all changes:
  - Metadata changes = minor version.
  - Data changes = major version.

1.3 Publish a Collection

1.3.1 Step 1: Log In & Create a Draft

- *Log in*.
- (Optional) Navigate to *a collection*.
- Click "Add Data" → "New Dataverse".

Note: If you don't see the "Add Data" button, contact your repository support team.

1.3.2 Step 2: Enter Basic Metadata & Settings

- Fill in the required *metadata* fields
- Select metadata settings.
- Click "Create Dataverse" at the bottom to save your draft collection.

1.3.3 Step 3: Publish Your Collection

Note: once published, easy deletion of a collection is no longer possible.

- Click "Publish" (top right).

1.3.4 Choose Look & Feel (optional)

- Click "Edit" → "Theme + Widgets".
- Select *theme settings*.
- Click "Save Changes".

1.3.5 Set Permissions (optional)

- Click "Edit" → "Permissions".
- Under "Permissions", click "Edit Access" to *set general permissions*.
- (Optional) Add users or groups with specific permissions under "Users/Groups" by clicking "Assign Roles to Users/Groups".

1.3.6 Create Groups (optional)

- Click "Edit" → "Groups".
- Click "Create Group".
- Enter a Group Name and Group Identifier.
- Add users to this group using autofill in "Users/Groups".
- Click "Create Group".

1.3.7 Set Dataset Templates (optional)

- Click "Edit" → "Dataset Templates".
- Click *Create Dataset Template*.
- Enter a template name.
- Add any template metadata in the metadata fields.
- Click "Save + Add Terms".
- *Choose a license* from the dropdown or select *Custom Dataset Terms*.

- Provide other relevant terms information
- Click "Save Dataset Template"

1.3.8 Set Dataset Guestbooks (optional)

- Click "Edit" → "Dataset Guestbooks".
- Click *Create Dataset Guestbook*.
- Enter a guestbook name.
- Add information to collect.
- Click "Create Dataset Guestbook".

CHAPTER

TWO

USER GUIDE

Contents:

2.1 Account Creation + Management

Contents:

- *Account Information*
  - *Account Log In Options*
  - *Create Account*
  - *Edit Account*
  - *Convert Account*
  - *Reset Account Password*
- *Remote Authentication*
  - *Institutional Log In*
    - *Create a Dataverse installation account using Institutional Log In*
    - *Convert your Dataverse installation account to use your Institutional Log In*
    - *Convert your Dataverse installation account away from your Institutional Log In*
    - *Troubleshooting Federated Institutional Log In*
  - *ORCID Log In*
    - *Create a Dataverse installation account using ORCID*
    - *Convert your Dataverse installation account to use ORCID for log in*
    - *Convert your Dataverse installation account away from ORCID for log in*
  - *Microsoft Azure AD, GitHub, and Google Log In*
- *Linking ORCID with Your Account Profile*
- *My Data*
- *Notifications*

- *API Token*
  - *What APIs Are and Why They Are Useful*
  - *How Your API Token Is Like a Password*
  - *How to Create Your API Token*
  - *How to Recreate Your API Token*
  - *Additional Information about API Tokens and Dataverse Software APIs*

2.1.1 Account Information

As a registered user, you can:

- Create your own Dataverse collection, if permitted, and customize it.
- Add datasets to Dataverse collections, if permitted.
- Contribute to existing datasets, if permitted.
- Request access to restricted files, if permitted.

Account Log In Options

The Dataverse installation has been configured for one or more of the following log in options:

- Username/Email and Password
- Institutional Log In
- ORCID
- Microsoft Azure AD
- GitHub
- Google

Please note that once you create your Dataverse installation account, it will be associated with only one of the log in options above.

The Institutional Log In, ORCID, Microsoft, GitHub, and Google options are described in more detail below under "Remote Authentication."

Create Account

To create a Dataverse installation account with the Username/Email log in option, use the "Sign Up" page. Fill out the fields, and then click the 'Create Account' button. Please note that the Username field does not support email addresses but will allow the following characters: a-Z, 0-9, _ (underscores), - (hyphens), and . (periods).

To create a Dataverse installation account associated with the log in option for your institution, ORCID, GitHub, or Google, use the "Log In" page and select one of the authentication providers. See the "Remote Authentication" section below for details.

Edit Account

1. To edit your account after you have logged in, click on your account name in the header on the right hand side and click on Account Information.
2. On the top right of your account page, click on the "Edit Account" button and from there you can select to edit either your Account Information or your Account Password.
3. Select "Save Changes" when you are done.

Please note that you cannot edit your account information within the Dataverse installation if you use the Institutional Log In option. Instead, you should contact your institution to change your name, email, etc. Once the change is made by your institution, it will be reflected in the Dataverse installation the next time you log in. Users of the Institutional Log In option are not required to verify their email address because the institution providing the email address is trusted.

Convert Account

If more than one log in option is available, you can convert from one to another and use the new option from now on.

If you are converting from the Email/Username log in option you will need to have your password ready to complete the conversion process. Click Log In, select the new log in option, and go through the log in process. When you return to the Dataverse installation, look for an option allowing you to convert and enter your password to complete the conversion. The section on Remote Authentication below has more specific information on converting to Institutional Log In, ORCID, GitHub, and Google.

If you need to perform any other conversion (i.e. from Google to GitHub), use the Support link at the top of the page for assistance.

Reset Account Password

Only Dataverse installation accounts using the Username/Email log in option have an associated password stored (securely!) in the Dataverse installation. If you cannot remember this password, click on Log In in the top right corner of any page and click the "Forgot Password?" link below where you would enter your username/email and password. Enter your email address and click "Submit Password Request" to receive an email with a link to reset your password.

Please note that if you have forgotten your username, you can use this same process to receive your username in an email.

2.1.2 Remote Authentication

Too many passwords? You can set up your Dataverse installation account to use log in credentials from one of the following remote authentication providers. This way, you can log in using your existing credentials from another service.

Institutional Log In

Institutional log in allows you to use your log in information for your university (e.g. HarvardKey at Harvard) to log in to your Dataverse installation account.

Create a Dataverse installation account using Institutional Log In

1. Click the "Log In" link in the navbar.
2. Select the "Your Institution" button under the "Other options" header
3. Using the dropdown menu, select your institution then click the Continue button to go to your institution's log in page.
4. After you put in your institutional credentials successfully, you will be brought back to the Dataverse installation to confirm your account information, and click "Create Account".
5. A username has been selected for you. You won't use this username to log in but it will appear next to your name when other users search for you to assign permissions within the system. To see what you username is, click on your name in the top right corner and click Account Information.

If you can't find your institution in a long list, you may need to request for it to be added to the "Research & Scholarship" category of an identity federation. See *Troubleshooting Federated Institutional Log In*.

If your institution is listed but you get login error ("eppn was null" or similar), it may mean your institution has declared itself part of the "Research & Scholarship" category of an identity federation but it is not releasing required attributes (often email) as it should. To resolve this, see *Troubleshooting Federated Institutional Log In*.

Convert your Dataverse installation account to use your Institutional Log In

If you already have a Dataverse installation account associated with the Username/Email log in option, but you want to convert it to use your institutional log in, you can easily do so as long as your account uses an email address from that institution.

1. Go to the Account Information page to confirm that your account email address is the same as your institutional email address. If not, you will need to update your Dataverse installation account to make them match.
2. Log out of the Dataverse installation.
3. Click the "Log In" link in the navbar.
4. Select the "Your Institution" button under the "Other options" header.
5. Using the dropdown menu, select your institution then click the Continue button to go to your institution's log in page.
6. After you put in your institutional credentials successfully, you will be brought back to the Dataverse installation to confirm your account information.
7. Enter your current password for your Dataverse installation account and click "Convert Account".
8. Now you have finished converting your Dataverse installation account to use your institutional log in.

Note that you cannot go through this conversion process if your Dataverse installation account associated with the Username/Email log in option has been deactivated.

Convert your Dataverse installation account away from your Institutional Log In

If you are leaving your institution and need to convert your Dataverse installation account to the Dataverse Username/Email log in option, you will need to contact support for the Dataverse installation you are using. On your account page, there is a link that will open a popup form to contact support for assistance.

Troubleshooting Federated Institutional Log In

Dataverse can be configured to allow institutional log in from a worldwide federation (eduGAIN) but for a successful log in, the following Research & Scholarship (R&S) attributes must be released:

- Shib-Identity-Provider
- eppn
- givenName
- sn
- email

If you have attempted to log in but are seeing an error such as `The SAML assertion for "eppn" was null`, you will need to contact the people who run the log in system (Identity Provider or IdP) for your organization and explain that the attributes above must be released. You can link them to this document, of course, as well as https://refeds.org/category/research-and-scholarship and *Identity Federation* in the Installation Guide.

Note that while Identity Providers (IdPs) who have joined R&S are required to release the attributes above to all Service Providers (SPs) who have joined R&S (Harvard Dataverse or UNC Dataverse, for example), for a successful login to a Dataverse installation, the IdP could decide to release attributes to just that individual installation. This is explained under *Specific Identity Provider(s)* in the Installation Guide.

ORCID Log In

You can set up your Dataverse installation account to allow you to log in using your ORCID credentials. ORCID® is an independent non-profit effort to provide an open registry of unique researcher identifiers and open services to link research activities and organizations to these identifiers. Learn more at orcid.org.

Create a Dataverse installation account using ORCID

1. Click the "Log In" link in the navbar.
2. Click the "ORCID" button under the "Other options" header.
3. Click the "Log In with ORCID" button to go to the ORCID website.
4. If you do not already have an ORCID account, you can create one on this page. If you already have an ORCID account, click on "Sign in" and then enter your login under the "Personal account" tab.
5. After you put in your ORCID credentials successfully, you will be brought back to Dataverse installation to confirm the creation of your Dataverse installation account. If your ORCID account's privacy settings permit it, the email address you've linked to your ORCID account will be suggested to you as an option. You can use this email if you like, or you can use any other email you might wish. If you have entered employment information within your ORCID account, the name of your employer will be suggested for the "Affiliation" field.

Convert your Dataverse installation account to use ORCID for log in

If you already have a Dataverse installation account associated with the Username/Email log in option, but you want to convert it to use ORCID for log in, follow these steps:

1. Log out of the Dataverse installation.
2. Click the "Log In" link in the navbar.
3. Click the "ORCID" button under the "Other options" header.
4. Click the "Log In with ORCID" button to go to the ORCID website.
5. If you do not already have an ORCID account, you can create one on this page. If you already have an ORCID account, click on "Sign in" and then enter your login under the "Personal account" tab.
6. After you put in your ORCID credentials successfully, you will be brought back to the Dataverse installation. Click the "convert your account" link.
7. Enter your username and password for your Dataverse installation account and click "Convert Account".
8. Now you have finished converting your Dataverse installation account to use ORCID for log in.

Note that you cannot go through this conversion process if your Dataverse installation account associated with the Username/Email log in option has been deactivated.

Convert your Dataverse installation account away from ORCID for log in

If you don't want to log in to the Dataverse installation using ORCID any more, you will want to convert your Dataverse installation account to the Dataverse installation Username/Email log in option. To do this, you will need to contact support for the Dataverse installation you are using. On your account page, there is a link that will open a popup form to contact support for assistance.

Microsoft Azure AD, GitHub, and Google Log In

You can also convert your Dataverse installation account to use authentication provided by GitHub, Microsoft, or Google. These options may be found in the "Other options" section of the log in page, and function similarly to how ORCID is outlined above. If you would like to convert your account away from using one of these services for log in, then you can follow the same steps as listed above for converting away from the ORCID log in.

2.1.3 Linking ORCID with Your Account Profile

If you login using ORCID, Dataverse will add the link to your ORCID account in your account profile and, when you create datasets, will automatically add you, with your ORCID, as an author.

If you login via other methods, you can add a link to your ORCID account as you create an account or later via the "Account Information" page. As when using ORCID login, you will be redirected to the ORCID website to log in there and allow the connection with Dataverse. Once you've done that, the link to your ORCID will be shown in the Account Information page and your ORCID will be added as your identifier when you create datasets (exactly the same as if you had logged in via ORCID).

Note that the ability to login via ORCID (or other providers) and the ability to link to your ORCID profile are separate configuration options *available* to Dataverse administrators.

2.1.4 My Data

The My Data section of your account page displays a listing of all the Dataverse collections, datasets, and files you have either created, uploaded or that you have a role assigned on. You are able to filter through all the Dataverse collections, datasets, and files listed on your My Data page using the filter box. You may also use the facets on the left side to only view a specific Publication Status or Role.

Note

If you see unexpected Dataverse collections or datasets in your My Data page, it might be because someone has assigned your account a role on those Dataverse collections or datasets. For example, some institutions automatically assign the "File Downloader" role on their datasets to all accounts using their institutional login.

You can use the Add Data button to create a new Dataverse collection or dataset. By default, the new Dataverse collection or dataset will be created in the root Dataverse collection, but from the create form you can use the Host Dataverse collection dropdown menu to choose a different Dataverse collection, for which you have the proper access privileges. However, you will not be able to change this selection after you create your Dataverse collection or dataset.

2.1.5 Notifications

Notifications appear in the notifications tab on your account page and are also displayed as a number next to your account name. You also receive notifications via email.

If your admin has enabled the option to change the notification settings, you will find an overview of the notification and email settings in the notifications tab. There, you can select which notifications and/or emails you wish to receive. If certain notification or email options are greyed out, you can't change the setting for this notification because the admin has set these as never to be muted by the user. You control the in-app and the email notifications separately in the two lists.

You will typically receive a notification or email when:

- You've created your account.
- You've created a Dataverse collection or added a dataset.
- Another Dataverse installation user has requested access to restricted files in a dataset that you published. (If you submitted your dataset for review, and it was published by a curator, the curators of the Dataverse collection that contains your dataset will get a notification about requests to access your restricted files.)
- A file in one of your datasets has finished the ingest process.

There are other notification types that you can receive, e.g., notification on granted roles, API key generation, etc. These types of notifications are less common and are not described here. Some other notifications are limited to specific roles. For example, if the installation has a curation workflow, reviewers get notified when a new dataset is submitted for review.

Notifications will only be emailed once, even if you haven't read the in-app notification.

It's possible to manage notifications via API. See *Notifications* in the API Guide.

2.1.6 API Token

What APIs Are and Why They Are Useful

API stands for "Application Programming Interface" and the Dataverse Software APIs allow you to take advantage of integrations with other software that may have been set up by admins of your Dataverse installation. See the *External Tools* and *Integrations* sections of the Admin Guide for examples of software that is commonly integrated with a Dataverse installation.

Additionally, if you are willing to write a little code (or find someone to write it for you), APIs provide a way to automate parts of your workflow. See the *Getting Started with APIs* section of the API Guide for details.

How Your API Token Is Like a Password

In many cases, such as when depositing data, an API Token is required to interact with Dataverse Software APIs. The word "token" indicates a series of letters and numbers such as `c6527048-5bdc-48b0-a1d5-ed1b62c8113b`. Anyone who has your API Token can add and delete data as you so you should treat it with the same care as a password.

How to Create Your API Token

To create your API token, click on your account name in the navbar, then select "API Token" from the dropdown menu. In this tab, click "Create Token".

How to Recreate Your API Token

If your API Token becomes compromised or has expired, click on your account name in the navbar, then select "API Token" from the dropdown menu. In this tab, click "Recreate Token".

Additional Information about API Tokens and Dataverse Software APIs

The Dataverse Software APIs are documented in the *API Guide* but the following sections may be of particular interest:

- *Getting Started with APIs*
- *API Tokens and Authentication*
- *Frequently Asked Questions*

2.2 Finding and Using Data

Contents:

- *Finding Data*
  - *Basic Search*
    - *Sorting and Viewing Search Results*
  - *Advanced Search*
  - *Geospatial Search*

  - *Browsing a Dataverse Installation*
  - *Saved Search*
- *Using Data*
  - *View Dataverse Collections + Datasets*
  - *View Files*
  - *File Search within Datasets*
  - *Tree View*
  - *Cite Data*
  - *Download Files*
    - *Tabular Data*
    - *Downloading via URL*
    - *Downloading a Dataverse File Package via URL*
  - *Explore Data*
  - *Exploratory Data Analysis Using Differentially Private Metadata (Experimental)*

2.2.1 Finding Data

Without logging in, users can browse a Dataverse installation and search for Dataverse collections, datasets, and files, view dataset descriptions and files for published datasets, and subset, analyze, and visualize data for published (restricted & not restricted) data files. To view an unpublished Dataverse collection, dataset, or file, a user will need to be given permission from that Dataverse installation's administrator to access it.

A user can search within a specific Dataverse collection for the Dataverse collections, datasets, and files it contains by using the search bar and facets displayed on that Dataverse collection's page.

Basic Search

You can search the entire contents of the Dataverse installation, including Dataverse collections, datasets, and files. You can access the search by clicking the "Search" button in the header of every page. The search bar accepts search terms, queries, or exact phrases (in quotations).

Sorting and Viewing Search Results

Facets: to the left of the search results, there are several facets a user can click on to narrow the number of results displayed.

- Choosing a facet: to choose a facet to narrow your results by, click on that facet.
- Removing a facet: A chosen facet can be removed by clicking on the X on it, either in the facets panel to the left, or above the results.
- Viewing more or fewer facets: Each category in the facets panel lists the top 5 most common facets from that category. To view more, click on "More…" in the bottom right of that category. Once you've chosen to see more, an option to view less will appear in the bottom left of the facet.

Result cards: after entering a search term or query, result cards that match your term or query appear underneath the search bar and to the right of the facets.

- Relevancy of results: each result card shows which metadata fields match the search query or term you entered into the search bar, with the matching term or query bolded. If the search term or query was found in the title or name of the Dataverse collection, dataset, or file, the search term or query will be bolded within it.

Other basic search features:

- Sorting results: search results can be sorted by name (A-Z or Z-A), by date (newest or oldest), or by relevancy of results. The sort button can be found above the search results, in the top right.
- Bookmarkable URLs: search URLs can be copied and sent to a fellow researcher, or can be bookmarked for future sessions.

Advanced Search

To perform an advanced search, click the "Advanced Search" link next to the search bar. There you will have the ability to enter search terms for Dataverse collections, dataset metadata (citation and domain-specific), and file-level metadata. If you are searching for tabular data files you can also search at the variable level for name and label. To find out more about what each field searches, hover over the field name for a detailed description of the field.

Geospatial Search

Geospatial search is available from the *Search API* (look for "geo" parameters). The metadata fields that are geospatially indexed are "West Longitude", "East Longitude", "North Latitude", and "South Latitude" from the "Geographic Bounding Box" field in the "Geospatial Metadata" block.

Browsing a Dataverse Installation

In a Dataverse installation, browsing is the default view when a user hasn't begun a search on the root Dataverse collection page or on a specific Dataverse collection's page. When browsing, only Dataverse collections and datasets appear in the results list and the results can be sorted by Name (A-Z or Z-A) and by Newest or Oldest. You can toggle the "Files" facet on the left to include files in the results list.

Saved Search

Saved Search is currently an experimental feature only available to superusers. Please see *Saved Search* in the API Guide for more information.

2.2.2 Using Data

View Dataverse Collections + Datasets

After performing a search and finding the Dataverse collection or dataset you are looking for, click on the name of the Dataverse collection or dataset or on the thumbnail image to be taken to the page for that Dataverse collection or dataset. Once on a Dataverse collection page, you can view the Dataverse collections, datasets, and files within that Dataverse collection.

Once on a dataset page, you will see the title, citation, description, and several other fields, as well as a button to email the dataset contact and a button to share the dataset on social media. Below that information, the files, metadata, terms of use, and version information for the dataset are available.

View Files

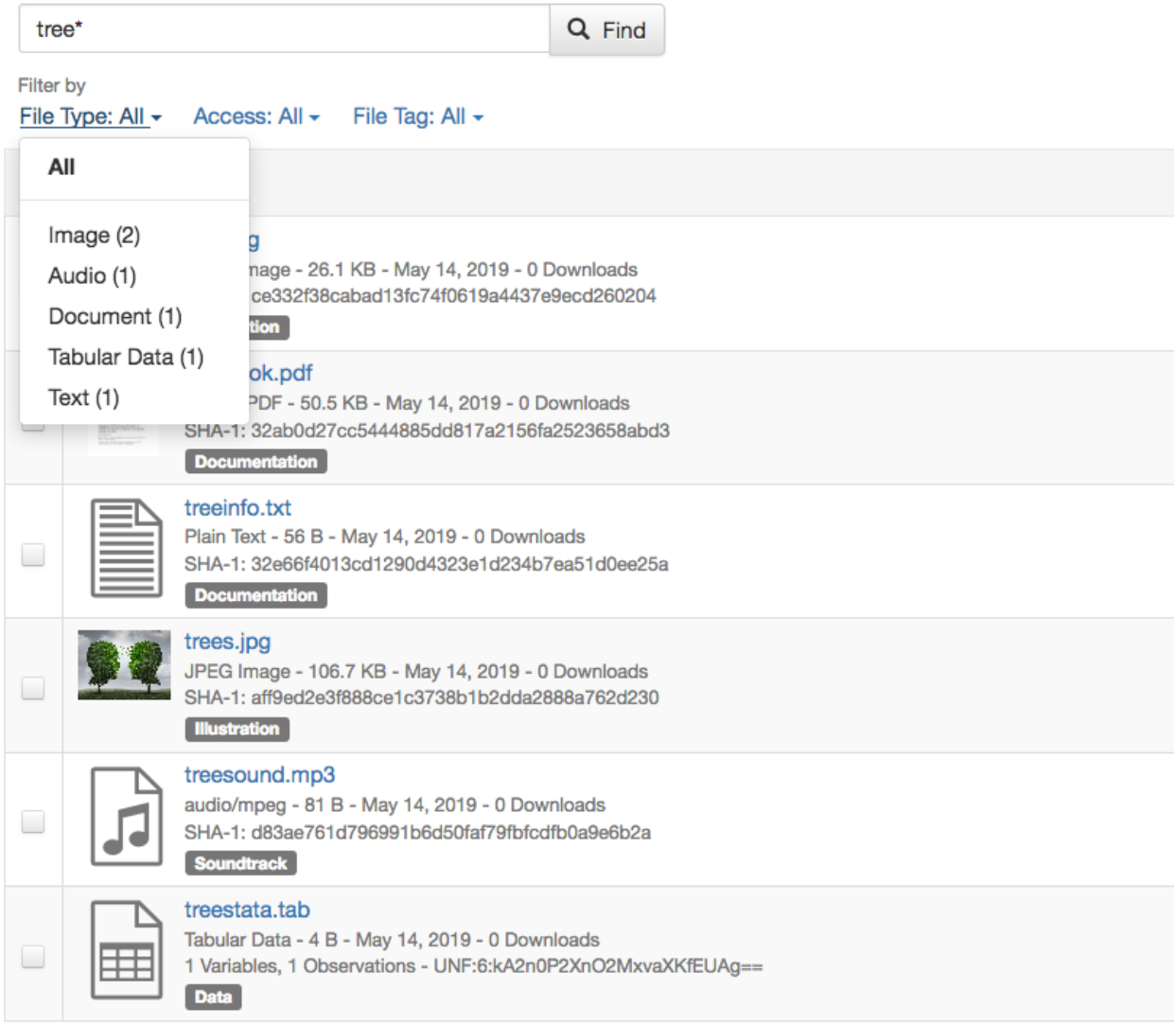
Files in a Dataverse installation each have their own landing page that can be reached through the search results or through the Files table on their parent dataset's page. The dataset page and file page offer much the same functionality in terms of viewing and editing files, with a few small exceptions.

- In installations that have enabled support for persistent identifiers (PIDs) at the file level, the file page includes the file's DOI or handle, which can be found in the file citation and also under the Metadata tab.
- Previewers for several common file types are available and can be added by installation administrators.
- The file page's Versions tab gives you a version history that is more focused on the individual file rather than the dataset as a whole.

File Search within Datasets

Datasets containing multiple files offer a file search function. On the Dataset page, under the Files tab, you'll see a search bar you can use to locate an individual file. It searches within the filename and file description. Performing a search will filter the file table to list only files matching your search. After you perform a search, if you'd like to return to the full list of files, just perform an empty search.

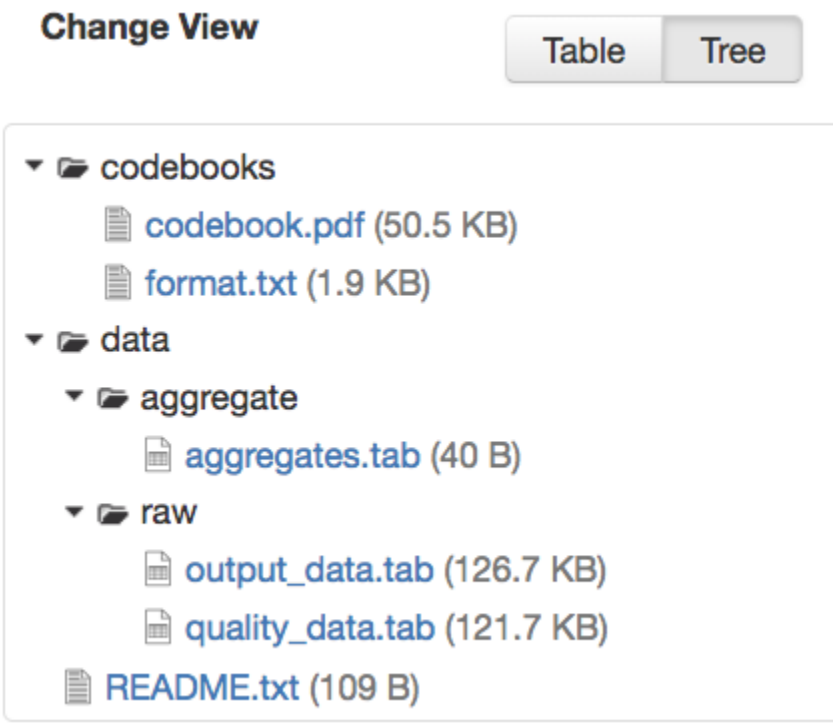
Under the search bar, you'll see file search facets you can use to filter the dataset's files by file type, access level, and file tags (see the example below).



(To provide these search facets, we rely on the Solr search engine. Only the latest published version and any draft version of each dataset are indexed in Solr. Because of that, facets cannot be offered for older versions of a dataset.)

Tree View

Files can be organized in one or more folders (directories) within a dataset. If the folder structure is defined, the Dataset Page will present an option for switching between the traditional table view, and the tree-like view showing folder and file hierarchy, as in the example below:



Cite Data

You can find the citation for the dataset at the top of the dataset page in a blue box. Additionally, there is a Cite Data button that offers the option to download the citation as EndNote XML, RIS Format, or BibTeX Format, or to cut/paste the citation in any of the 1000+ standard journal/society/other formats defined via the Citation Style Language.

In installations that have added file-level citations, you can find and download the file's citation on the file page in a similar manner.

Download Files

If you want to download all files in a dataset, you can click the Access Dataset dropdown on the dataset page and select one of the download options. The dataset's files will download in .zip format and will preserve any folder structure that the dataset owner had set up.

If you'd like to download a single file or some subset of the dataset's files, you can use the Files tab. To download more than one file at a time, select the files you would like to download and then click the Download button above the files. The dataset's files will download in .zip format and will preserve any folder structure that the dataset owner had set up.

You may also download a file from the Access File button on its file page or by *Downloading via URL* under the Metadata tab.

Tabular data files offer additional options: You can explore using any data exploration or visualization *External Tools* (if they have been enabled), or choose from a number of tabular-data-specific download options available.

Tabular Data

Ingested files can be downloaded in several different ways.

- The default option is to download a tab-separated-value file which is an easy and free standard to use.
- The original file, which may be in a proprietary format which requires special software
- RData format if the installation has configured this
- The variable metadata for the file in DDI format

Downloading via URL

The Dataverse installation displays a plaintext URL for the location of the file under the Metadata tab on the file page. This Download URL can be used to directly access the file via API (or in a web browser, if needed). When downloading larger files, in order to ensure a reliable, resumable download, we recommend using GNU Wget in a command line terminal or using a download manager software of your choice.

Certain files do not provide Download URLs for technical reasons: those that are restricted, have terms of use associated with them, or are part of a Dataverse collection with a guestbook enabled.

Downloading a Dataverse File Package via URL

Dataverse File Packages are typically used to represent extremely large files or bundles containing a large number of files. Dataverse File Packages are often too large to be reliably downloaded using a web browser. When you click to download a Dataverse File Package, instead of automatically initiating the download in your web browser, the Dataverse installation displays a plaintext URL for the location of the file. To ensure a reliable, resumable download, we recommend using GNU Wget in a command line terminal or using a download manager software of your choice. If you try to simply paste the URL into your web browser then the download may overwhelm your browser, resulting in an interrupted, timed out, or otherwise failed download.

Explore Data

Some file types and datasets offer data exploration options if external tools have been installed. The tools are described in the *External Tools* section of the Admin Guide.

Exploratory Data Analysis Using Differentially Private Metadata (Experimental)

Through an integration with tools from the OpenDP Project (opendp.org), the Dataverse Software offers an experimental workflow that allows a data depositor to create and deposit Differentially Private (DP) Metadata files, which can then be used for exploratory data analysis. This workflow allows researchers to view the DP metadata for a tabular file, determine whether or not the file contains useful information, and then make an informed decision about whether or not to request access to the original file.

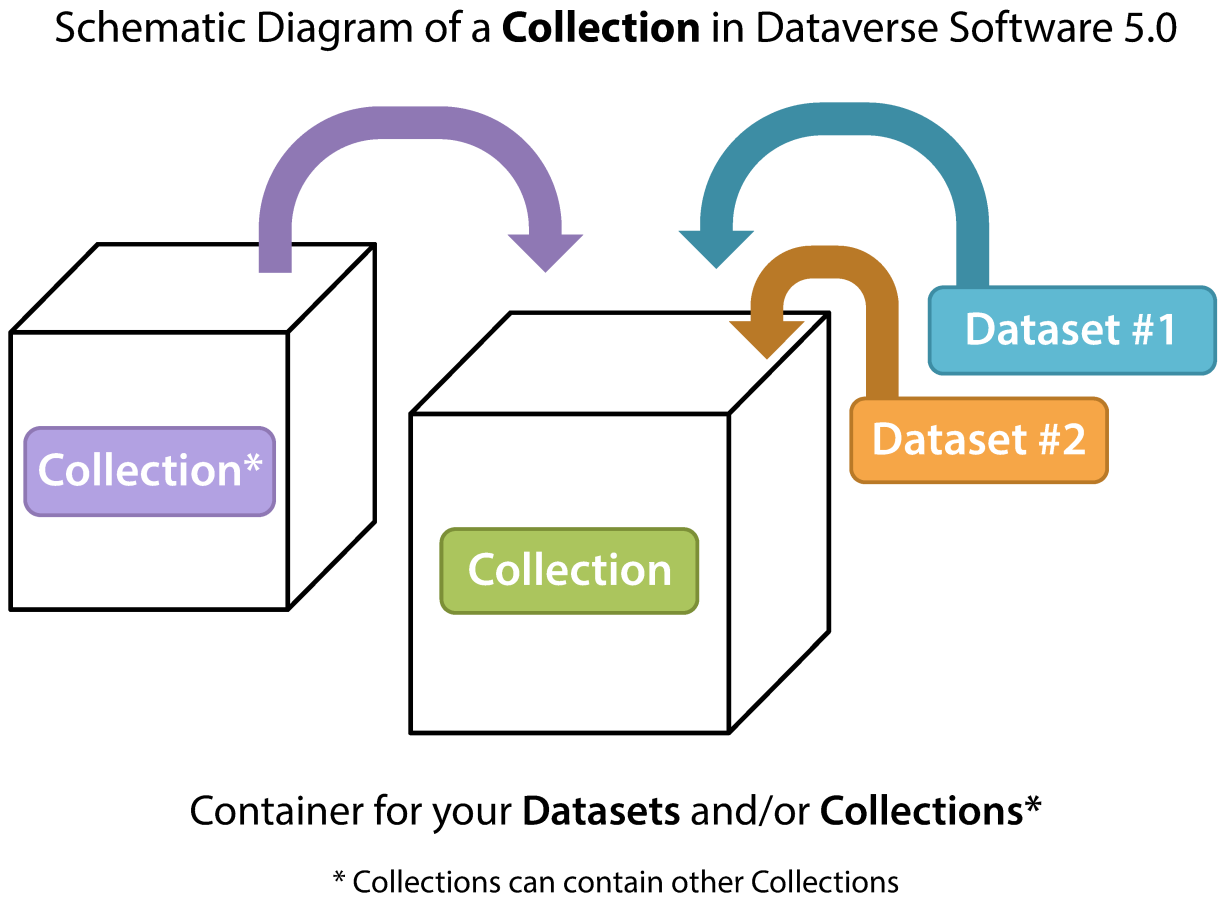
If the data depositor has made available DP metadata for one or more files in their dataset, these access options will appear on the access dropdown on both the Dataset Page and the File Page. These access options will be available even if a file is restricted. Three types of DP metadata will be available:

- .PDF
- .XML
- .JSON

For more information about how data depositors can enable access using the OpenDP tool, visit the *Dataset + File Management* section of the User Guide.

2.3 Dataverse Collection Management

A Dataverse collection is a container for datasets (research data, code, documentation, and metadata) and other Dataverse collections, which can be setup for individual researchers, departments, journals and organizations.



Once a user creates a Dataverse collection they, by default, become the administrator of that Dataverse collection. The Dataverse collection administrator has access to manage the settings described in this guide.

Contents:

- *Create a New Dataverse Collection*
- *Edit Dataverse Collection*
  - *General Information*
  - *Theme*
  - *Widgets*

        * *Dataverse Collection Search Box Widget*
        * *Dataverse Collection Listing Widget*
        * *Adding Widgets to an OpenScholar Website*
      - *Roles & Permissions*
        * *Setting Access Configurations*
        * *Assigning Roles to Users and Groups*
      - *Dataset Templates*
      - *Dataset Guestbooks*
      - *Featured Dataverse Collection*
  - *Dataset Linking*
  - *Dataverse Collection Linking*
  - *Publish Your Dataverse Collection*
  - *Locally FAIR*
      - *What Locally FAIR Means*
      - *Why Repositories Use It*
      - *How It Differs from Restricted Files*
      - *Who Can See Locally FAIR Content*
      - *How Can You Tell When Content is Locally FAIR?*
      - *Why is Locally FAIR Support "Experimental"*
      - *Things to Keep in Mind*
      - *Enabling Locally FAIR Support*
      - *Managing Locally FAIR Via API*

2.3.1 Create a New Dataverse Collection

Creating a Dataverse collection is easy but first you must be a registered user (see *Account Creation + Management*). Once you are logged in, click on the "Add Data" button, or "Add Data" navbar link and in the dropdown menu select "New Dataverse".

1. **Fill in the following fields on the "New Dataverse" page (required fields in the form are denoted by a red asterisk, next to the label):**
    - **Host Dataverse**: Select a Dataverse collection you would like to create the new Dataverse collection in, by default the root Dataverse collection is selected if you click the Add Data button from the root Dataverse collection, your My Data page, the Add Data link in the navbar, or the Add a dataset button on the custom homepage (if configured), but you can select any Dataverse collection for which you the proper permissions to create Dataverse collections.
    - **Dataverse Name**: Enter the name of your Dataverse collection.
    - **Affiliation**: Add any affiliation that can be associated with this particular Dataverse collection (e.g., project name, institute name, department name, journal name, etc). This is automatically filled out if you have added an affiliation for your user account.

- **Identifier**: This is an abbreviation, usually lower-case, that becomes part of the URL for the new Dataverse collection. Special characters (~,`, !, @, #, $, %, ^, &, and *) and spaces are not allowed. **Note**: if you change this field in the future, the URL for your Dataverse collection will change (http//.../'identifier'), which will break older links to the page.
- **Category**: Select a category that best describes the type of Dataverse collection this will be. For example, if this is a Dataverse collection for an individual researcher's datasets, select *Researcher*. If this is a Dataverse collection for an institution, select *Organization or Institution*.
- **Email**: This is the email address that will be used as the contact for this particular Dataverse collection. You can have more than one contact email address for your Dataverse collection.
- **Description**: Provide a description of this Dataverse collection. This will display on the landing page of your Dataverse collection and in the search result list. The description field supports certain HTML tags, if you'd like to format your text (<a>, <b>, <blockquote>, <br>, <code>, <del>, <dd>, <dl>, <dt>, , <hr>, <h1>-<h3>, <i>, <img>, <kbd>, <li>, <ol>, <p>, <pre>, <s>, <sup>, <sub>, , <strike>, <u>, <ul>).
- **Dataset Metadata Langauge**: (If enabled) Select which language should be used when entering dataset metadata, or leave that choice to dataset creators.
- **Guestbook Entry Option**: (If enabled) Select whether guestbooks are displayed when a user requests access to restricted file(s) or when they initiate a download.

2. **Choose the sets of Metadata Fields for datasets in this Dataverse collection:**
   - By default the metadata elements will be from the host Dataverse collection that this new Dataverse collection is created in.
   - The Dataverse Software offers metadata standards for multiple domains. To learn more about the metadata standards in the Dataverse Software please check out the *Appendix*.
   - Most metadata fields can be hidden or can be selected as required or optional. Some fields may be selected as conditionally required, depending on metadata options chosen for the Dataverse installation.
   - Selected metadata fields are also used to pick which metadata fields you would like to use for creating *Dataset Templates*: after you finish creating your Dataverse collection.
3. **Choose which metadata fields will be used as browse/search facets on your Dataverse collection:**
   - These facets will allow users browsing or searching your Dataverse collection to filter its contents according to the fields you have selected. For example, if you select "Subject" as a facet, users will be able to filter your Dataverse collection's contents by subject area.
   - By default, the facets that will appear on your Dataverse collection's landing page will be from the host Dataverse collection that this new Dataverse collection was created in, but you can add or remove facets from this default.
4. **Click the "Create Dataverse" button** and you're done!

2.3.2 Edit Dataverse Collection

To edit your Dataverse collection, navigate to your Dataverse collection's landing page and select the "Edit Dataverse" button, where you will be presented with the following editing options:

- *General Information*: edit name, affiliation, identifier, category, contact email, description, metadata fields, and browse/search facets for your Dataverse collection
- *Theme*: upload a logo for your Dataverse collection, add a link to your department or personal website, add a custom footer image, and select colors for your Dataverse collection in order to brand it
- *Widgets*: get code to add to your website to have your Dataverse collection display on it

- *Permissions*: give other users permissions to your Dataverse collection, i.e.-can edit datasets, and see which users already have which permissions for your Dataverse collection
- *Dataset Templates*: these are useful when you want to provide custom instructions on how to fill out fields or have several datasets that have the same information in multiple metadata fields that you would prefer not to have to keep manually typing in
- *Dataset Guestbooks*: allows you to collect data about who is downloading the files from your datasets
- *Featured Dataverse collections*: if you have one or more Dataverse collection, you can use this option to show them at the top of your Dataverse collection page to help others easily find interesting or important Dataverse collections
- **Delete Dataverse**: you are able to delete your Dataverse collection as long as it is not published and does not have any draft datasets

General Information

The General Information page is how you edit the information you filled in while creating your Dataverse collection. If you need to change or add a contact email address, this is the place to do it. Additionally, you can update the metadata elements used for datasets within the Dataverse collection, change which metadata fields are hidden, required, or optional, and update the facets you would like displayed for browsing the Dataverse collection. If you plan on using templates, you need to select the metadata fields on the General Information page.

Tip: The metadata fields you select as required will appear on the Create Dataset form when someone goes to add a dataset to the Dataverse collection.

Theme

The Theme features provides you with a way to customize the look of your Dataverse collection. You can:

- Inherit the theme from the parent Dataverse collection. This option is helpful if you'd like consistency across several Dataverse collections that all share the same parent.
- Add or update a logo image, which will appear at the top of your Dataverse collection.
- Add or update a thumbnail image, which will appear on featured dataverses of your Dataverse collection.
- Add or update a footer image, which will appear at the bottom of your Dataverse collection.
- Change the colors of the background, links, and text within the header of your Dataverse collection.
- Add or update the tagline for your Dataverse collection, which can provide more information about your organization, journal, institution, etc.
- Add a URL for a website that will be accessed when visitors click the tagline text.

Supported image types for logo images and footer images are JPEG, TIFF, or PNG and should be no larger than 500 KB. The maximum display size for an image file in a Dataverse collection's theme is 940 pixels wide by 120 pixels high.

Widgets

The Widgets feature provides you with code for you to put on your personal website to have your Dataverse collection displayed there. There are two types of Widgets for a Dataverse collection, a Dataverse collection Search Box widget and a Dataverse collection Listing widget. Once a Dataverse collection has been published, from the Widgets tab on the Dataverse collection's Theme + Widgets page, it is possible to copy the code snippets for the widget(s) you would like to add to your website. If you need to adjust the height of the widget on your website, you may do so by editing the *heightPx=500* parameter in the code snippet.

Dataverse Collection Search Box Widget

The Dataverse Collection Search Box Widget will add a search box to your website that is linked to your Dataverse collection. Users are directed to your Dataverse collection in a new browser window, to display the results for search terms entered in the input field.

Dataverse Collection Listing Widget

The Dataverse Collection Listing Widget provides a listing of all your Dataverse collections and datasets for users to browse, sort, filter and search. When someone clicks on a Dataverse collection or dataset in the widget, it displays the content in the widget on your website. They can download data files directly from the datasets within the widget. If a file is restricted, they will be directed to your Dataverse installation to log in, instead of logging in through the widget on your website.

Adding Widgets to an OpenScholar Website

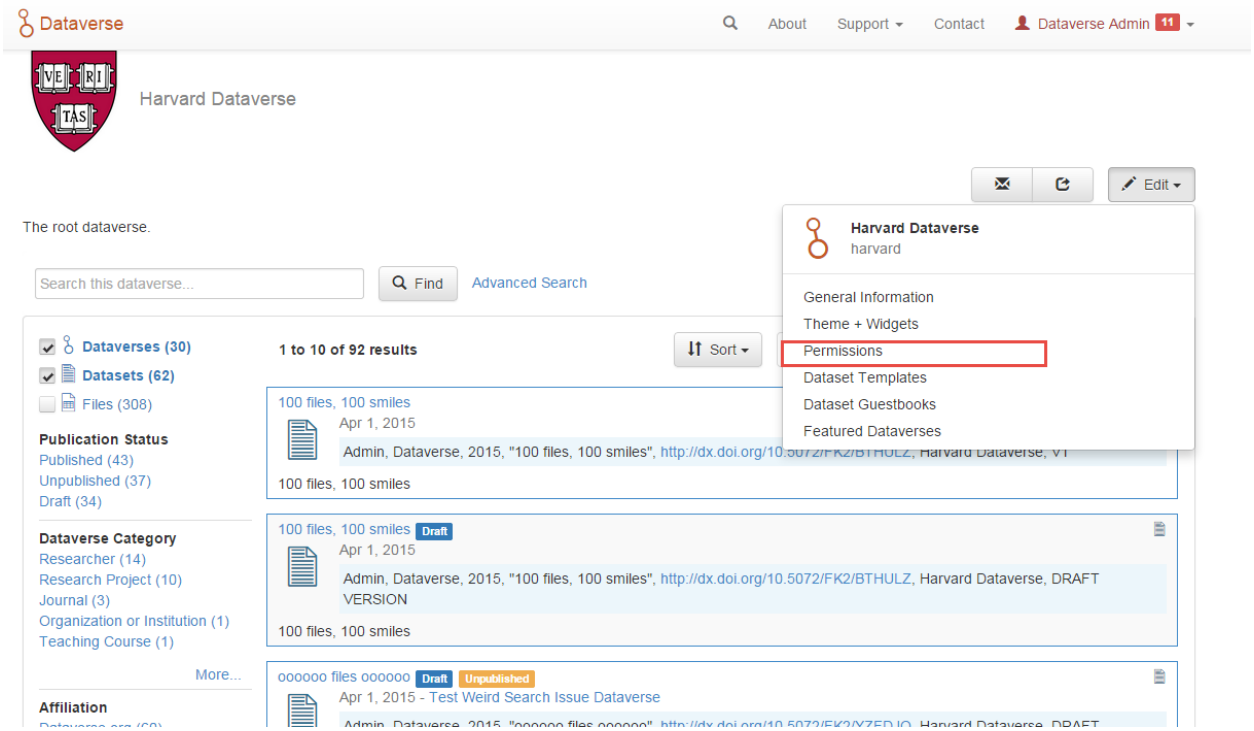
1. Log in to your OpenScholar website
2. Either build a new page or navigate to the page you would like to use to show the Dataverse collection widgets.
3. Click on the Settings Cog and select Layout
4. At the top right, select Add New Widget and under Misc. you will see the Dataverse Collection Search Box and the Dataverse Collection Listing Widgets. Click on the widget you would like to add, fill out the form, and then drag it to where you would like it to display in the page.

Roles & Permissions

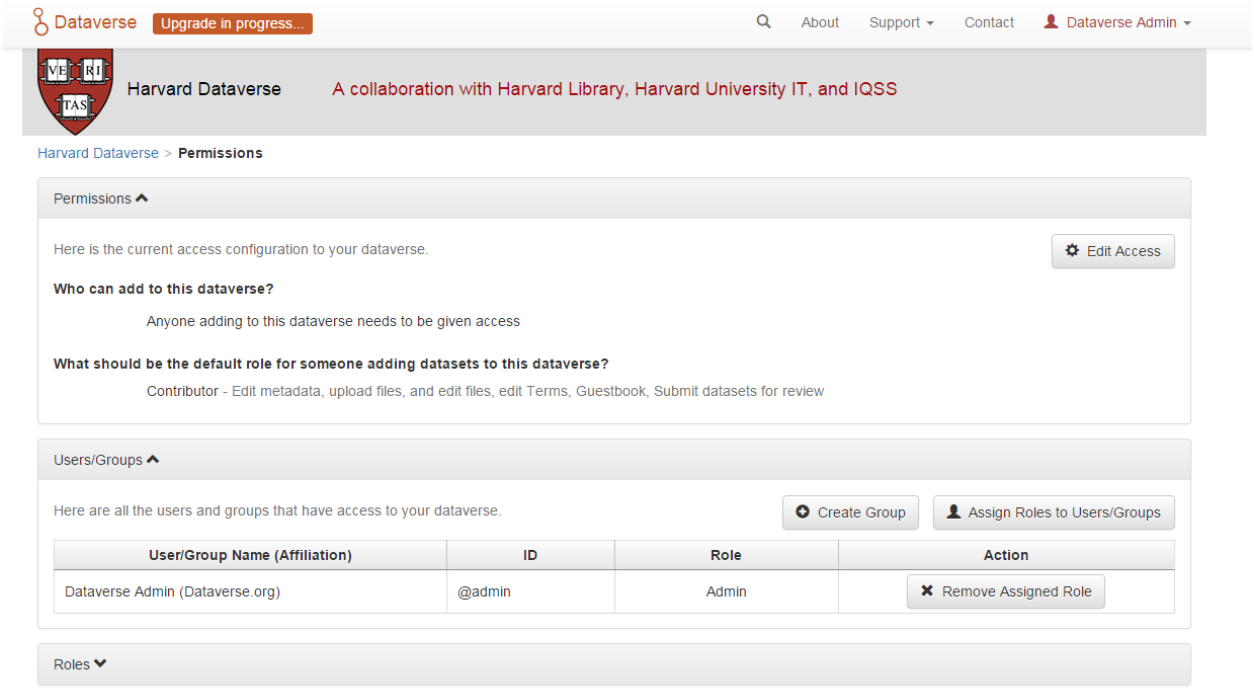
Dataverse installation user accounts can be granted roles that define which actions they are allowed to take on specific Dataverse collections, datasets, and/or files. Each role comes with a set of permissions, which define the specific actions that users may take.

Roles and permissions may also be granted to groups. Groups can be defined as a collection of Dataverse installation user accounts, a collection of IP addresses (e.g. all users of a library's computers), or a collection of all users who log in using a particular institutional login (e.g. everyone who logs in with a particular university's account credentials).

The user who creates a dataverse is given the "Admin" role on that dataverse (see *Root Dataverse Collection Permissions*). Admins of a Dataverse collection can assign roles and permissions to the users of that Dataverse collection. If you are an admin on a Dataverse collection, then you will find the link to the Permissions page under the Edit dropdown on the Dataverse collection page.



Clicking on Permissions will bring you to this page:



When you access a Dataverse collection's permissions page, you will see three or four sections:

Permissions: Here you can decide the requirements that determine which types of users can add datasets and sub Dataverse collections to your Dataverse collection, and what permissions they'll be granted when they do so.

Users/Groups: Here you can assign roles to specific users or groups, determining which actions they are permitted to take on your Dataverse collection. You can also reference a list of all users who have roles assigned to them for your

Dataverse collection and remove their roles if you please.

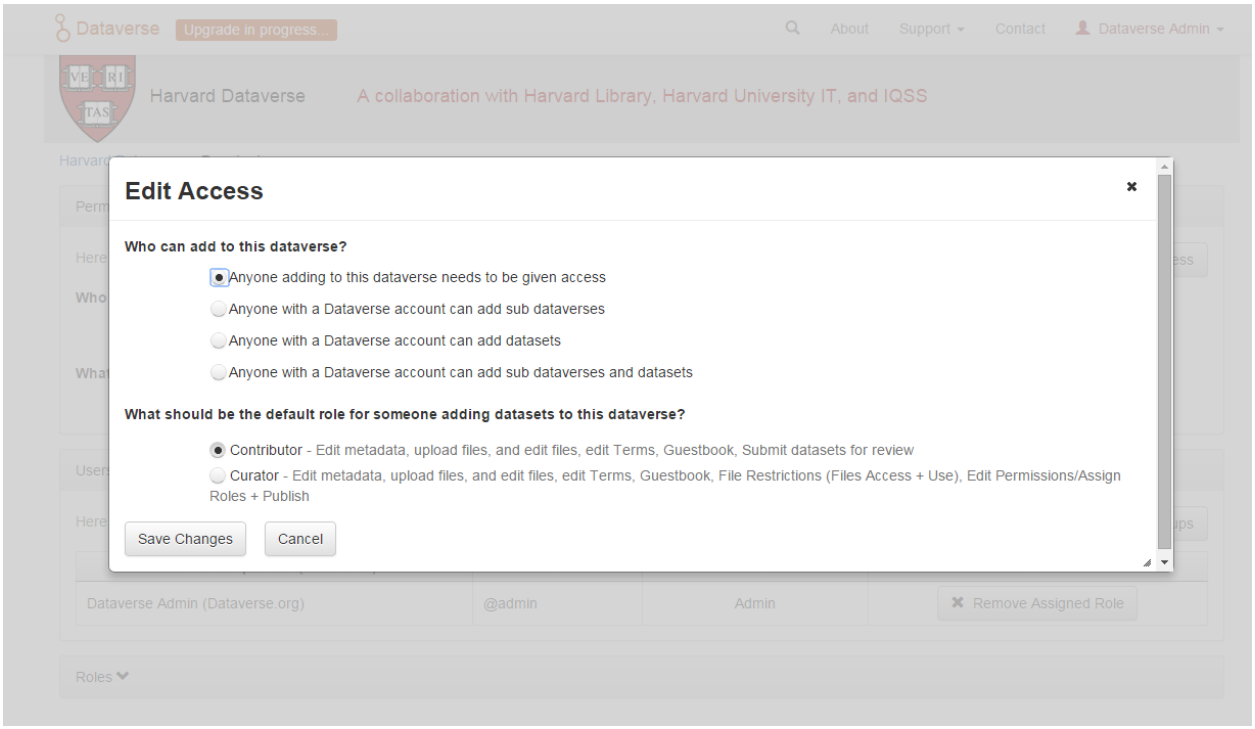
Roles: Here you can reference a full list of roles that can be assigned to users of your Dataverse collection. Each role lists the permissions that it offers.

Role Assignment History If enabled, you'll be able to see the history of when roles have been assigned and revoked and by whom.

Please note that even on a newly created Dataverse collection, you may see user and groups have already been granted role(s) if your installation has `:InheritParentRoleAssignments` set. For more on this setting, see the *Configuration* section of the Installation Guide.

Setting Access Configurations

Under the Permissions tab, you can click the "Edit Access" button to open a box where you can add to your Dataverse collection and what permissions are granted to those who add to your Dataverse collection.



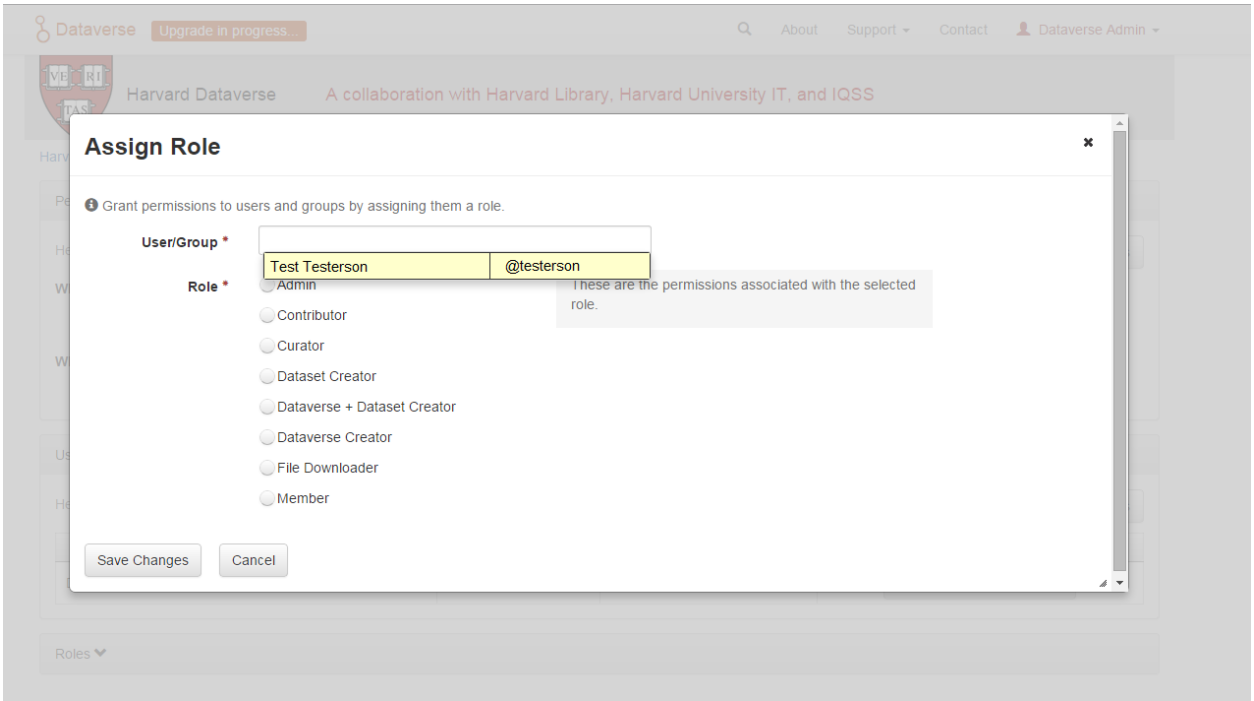
The first question on this page allows you to determine how open your Dataverse collection is to new additions - you can set whether or not the entire userbase (all logged in users) has the ability to add datasets or sub Dataverse collections to your Dataverse collection.

The second question on this page allows you to choose the role (and thus the permissions) granted to users who add a dataset to your Dataverse collection. The role you select will be automatically granted to any user who creates a dataset on your Dataverse collection, on that dataset, at the moment that they create it. The role the user is given determines their permissions for the dataset they've created. The key difference between the two roles is that curators can publish their own datasets, while contributors must submit the dataset to be reviewed before publication. Additionally, curators can manage dataset permissions. Note that this setting does not retroactively apply roles to users who have previously added datasets to your Dataverse collection; it only applies to users adding new datasets going forward.

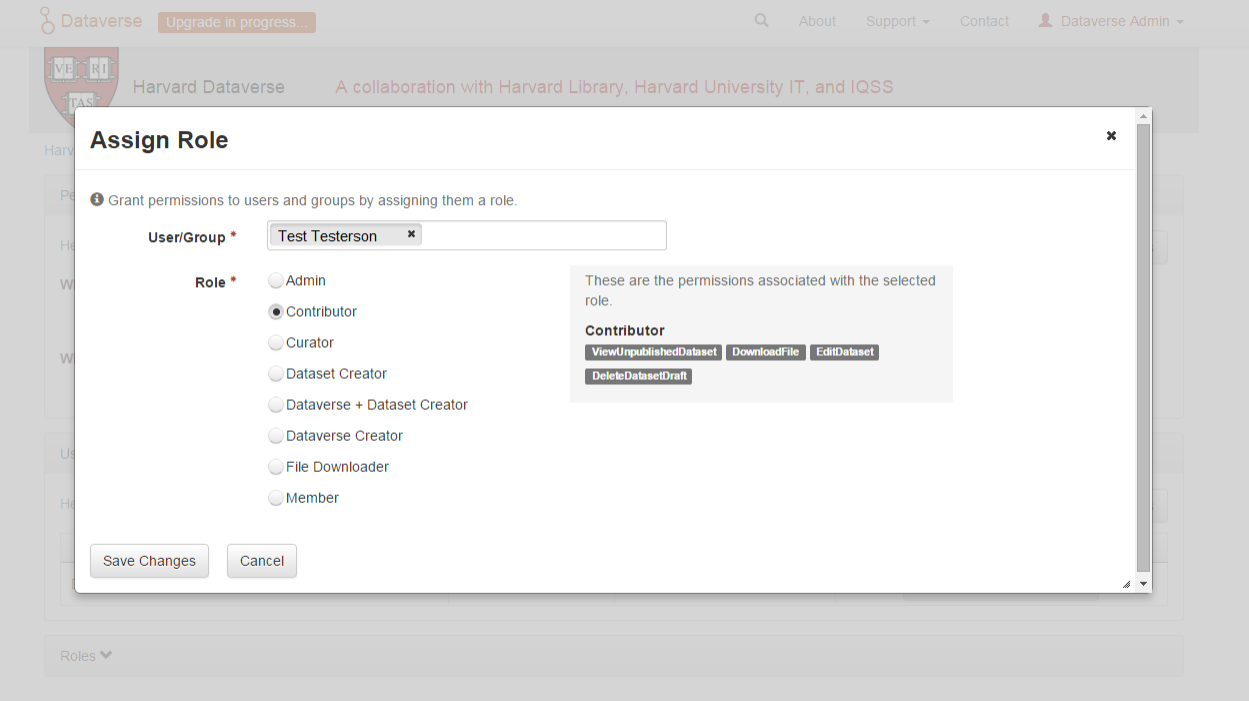
Both of these settings can be changed at any time.

Assigning Roles to Users and Groups

Under the Users/Groups tab, you can add, edit, or remove the roles granted to users and groups on your Dataverse collection. A role is a set of permissions granted to a user or group when they're using your Dataverse collection. For example, giving your research assistant the "Contributor" role would give them the following self-explanatory permissions on your Dataverse collection and all datasets within your Dataverse collection: "ViewUnpublishedDataset", "DownloadFile", "EditDataset", and "DeleteDatasetDraft". They would, however, lack the "PublishDataset" permission, and thus would be unable to publish datasets on your Dataverse collection. If you wanted to give them that permission, you would give them a role with that permission, like the Curator role. Users and groups can hold multiple roles at the same time if needed. Roles can be removed at any time. All roles and their associated permissions are listed under the "Roles" tab of the same page.



Note that the Dataset Creator role and Contributor role are sometimes confused. The Dataset Creator role is assigned at the Dataverse collection level and allows a user to create new datasets in that Dataverse collection. The Contributor role can be assigned at the dataset level, granting a user the ability to edit *that specific* dataset. Alternatively, the Contributor role can be assigned at the Dataverse collection level, granting the user the ability to edit *all* datasets in that Dataverse collection.



Note: If you need to assign a role to ALL user accounts in a Dataverse installation, you can assign the role to the ":authenticated-users" group.

Dataset Templates

Templates are useful when you want to provide custom instructions on how to fill out a field, have several datasets that have the same information in multiple metadata fields that you would prefer not to have to keep manually typing in, or if you want to use a custom set of Terms of Use and Access for multiple datasets in a Dataverse collection. In Dataverse Software 4.0+, templates are created at the Dataverse collection level, can be deleted (so it does not show for future datasets), set to default (not required), or can be copied so you do not have to start over when creating a new template with similar metadata from another template. When a template is deleted, it does not impact the datasets that have used the template already.

How do you create a template?

1. Navigate to your Dataverse collection, click on the Edit Dataverse button and select Dataset Templates.
2. Once you have clicked on Dataset Templates, you will be brought to the Dataset Templates page. On this page, you can 1) decide to use the dataset templates from your parent Dataverse collection 2) create a new dataset template or 3) do both.
3. Click on the Create Dataset Template to get started. You will see that the template is the same as the create dataset page with an additional field at the top of the page to add a name for the template.
4. To add custom instructions, click on ''(None - click to add)" and enter the instructions you wish users to see. If you wish to edit existing instructions, click on them to make the text editable.
5. After adding information into the metadata fields you have information for and clicking Save and Add Terms, you will be brought to the page where you can add custom Terms of Use and Access. If you do not need custom Terms of Use and Access, click the Save Dataset Template, and only the metadata fields will be saved.
6. After clicking Save Dataset Template, you will be brought back to the Manage Dataset Templates page and should see your template listed there now with the make default, edit, view, or delete options.

7. A Dataverse collection does not have to have a default template and users can select which template they would like to use while on the Create Dataset page.
8. You can also click on the View button on the Manage Dataset Templates page to see what metadata fields have information filled in.

* Please note that the ability to choose which metadata fields are hidden, required, or optional is done on the General Information page for the Dataverse collection.

Dataset Guestbooks

Guestbooks allow you to collect data about who is downloading the files from your datasets. You can decide to collect account information (username, given name & last name, affiliation, etc.) as well as create custom questions (e.g., What do you plan to use this data for?). You are also able to download the data collected from the enabled guestbooks as CSV files to store and use outside of the Dataverse installation.

How do you create a guestbook?

1. After creating a Dataverse collection, click on the "Edit Dataverse" button and select "Dataset Guestbooks".
2. To create a new guestbook, click the "Create Dataset Guestbook" button on the right side of the page.
3. Name the guestbook, determine the account information that you would like to be required (all account information fields show when someone downloads a file), and then add custom questions (can be required or not required), if desired.
4. Click the "Create Dataset Guestbook" button once you have finished.

What can you do with a guestbook? After creating a guestbook, you will notice there are several options for a guestbook that appear in the list of guestbooks.

- From the dataset page, you can select a guestbook by clicking "Terms" and then "Edit Terms Requirements", unless the guestbook is disabled.
- From the "Manage Dataset Guestbooks" page, there are options to view, copy, edit, disable, or delete a guestbook. There are also options to download or view responses. By default, guestbooks inherited from the parent Dataverse collection will appear. If you do not want to use or see those guestbooks, uncheck the checkbox that says "Include Guestbooks from [Parent]".
- Note that it is also possible to download guestbook responses via API. See *Retrieve Guestbook Responses for a Dataverse Collection* for details.

Featured Dataverse Collection

Featured Dataverse collections is a way to display sub Dataverse collections in your Dataverse collection that you want to feature for people to easily see when they visit your Dataverse collection.

Click on Featured Dataverse Collections and a pop up will appear. Select which Dataverse subcollections you would like to have appear.

Note: Featured Dataverse collections can only be used with published Dataverse collections.

2.3.3 Dataset Linking

Dataset linking allows a Dataverse collection owner to "link" their Dataverse collection to a dataset that exists outside of that Dataverse collection, so it appears in the Dataverse collection's list of contents without actually *being* in that Dataverse collection. You can link other users' datasets to your Dataverse collection, but that does not transfer editing or other special permissions to you. The linked dataset will still be under the original user's control.

For example, researchers working on a collaborative study across institutions can each link their own individual institutional Dataverse collections to the one collaborative dataset, making it easier for interested parties from each institution to find the study.

In order to link a dataset, you will need your account to have the "Link Dataset" permission on the Dataverse collection that is doing the linking. If you created the Dataverse collection then you should have this permission already, but if not then you will need to ask the admin of that Dataverse collection to assign that permission to your account. You do not need any special permissions on the dataset being linked.

To link a dataset to your Dataverse collection, you must navigate to that dataset and click the white "Link" button in the upper-right corner of the dataset page. This will open up a window where you can type in the name of the Dataverse collection that you would like to link the dataset to. Select your Dataverse collection and click the save button. This will establish the link, and the dataset will now appear under your Dataverse collection.

A draft dataset can be linked to other Dataverse collections. It will only become publicly visible in the linked collection(s) after it has been published. To publish the dataset, your account must have the "Publish Dataset" permission for the Dataverse collection in which the dataset was originally created. Permissions in the linked Dataverse collections do not apply.

There is currently no way to remove established links in the UI. If you need to remove a link between a Dataverse collection and a dataset, please contact the support team for the Dataverse installation you are using (see the *Unlink a Dataset* section of the Admin Guide for more information).

2.3.4 Dataverse Collection Linking

Similarly to dataset linking, Dataverse collection linking allows a Dataverse collection owner to "link" their Dataverse collection to another Dataverse collection, so the Dataverse collection being linked will appear in the linking Dataverse collection's list of contents without actually *being* in that Dataverse collection. Currently, the ability to link a Dataverse collection to another Dataverse collection is a superuser only feature.

If you need to have a Dataverse collection linked to your Dataverse collection, please contact the support team for the Dataverse installation you are using.

2.3.5 Publish Your Dataverse Collection

Once your Dataverse collection is ready to go public, go to your Dataverse collection page, click on the "Publish" button on the right hand side of the page. A pop-up will appear to confirm that you are ready to actually Publish, since once a Dataverse collection is made public, it can no longer be unpublished.

2.3.6 Locally FAIR

Locally FAIR describes content that is managed according to FAIR principles (Findable, Accessible, Interoperable, and Reusable) within a defined local or organizational community rather than for the public internet as a whole.

Dataverse now has optional, experimental support for managing Locally FAIR collections.

In a typical public Dataverse installation, published dataset metadata is visible to everyone, even if the dataset's files themselves may be embargoed or restricted. Locally FAIR support extends this model by allowing some collections, and the published datasets within them to remain visible only to designated users or groups. This makes it possible for a single Dataverse installation to support both:

- public, globally discoverable content; and
- organizational content whose existence and metadata are only be visible to authorized users.

The rationale for making some content Locally FAIR can vary. Locally FAIR content can include:

- sensitive research collections;
- institution-only datasets;
- datasets that should not be accessible to bots that may not adhere to the dataset license and terms, and
- projects under contractual or policy restrictions;

Dataverse's Locally FAIR mechanism is appropriate for repositories that will house at least some data whose metadata should only be visible to organizational members. The decision to make data Locally FAIR is managed at the collection level and repositories can have both FAIR and Locally FAIR content.

What Locally FAIR Means

Locally FAIR content is intended to be FAIR within a particular community.

That means:

- **Findable** Data is easy to locate for both humans and machines, when authorized. Locally FAIR datasets (and files if configured) have persistent identifiers, but do not use DOIs which are publicly searchable.
- **Accessible** Data is retrievable through standardized protocols. Authorized users can use Dataverse's standard user interface and API calls to access Locally FAIR content in the same way they do with any published data.
- **Interoperable** Data should be compatible with other datasets and systems. Locally FAIR datasets in Dataverse use the same standard metadata blocks as for public content and files undergo the same ingest process, use the same previewers and tools, etc.
- **Reusable** Data should be well-described and licensed in a way that allows others to use it for future research. The licenses and terms on Locally FAIR content make it clear how and when the data can be re-used.

Why Repositories Use It

Without Locally FAIR support, repositories may need separate Dataverse installations to separate public and organization-only content.

How It Differs from Restricted Files

Restricting or *embargoing* files limits access to the file contents, but in a standard public repository the dataset's published metadata, including the list of files, would still be visible. If a dataset allows requests for file access, anyone can request access, even if the dataset's license or terms limit access to specific groups.

Locally FAIR goes further. Locally FAIR collections and datasets do not appear in content listings or search results for unauthorized users, nor can the collection/dataset/file page be viewed. API access is also blocked for unauthorized access.

Who Can See Locally FAIR Content

Visibility is determined by superusers and is managed at the collection level. Access can be granted to any groups or users defined in Dataverse - the same groups/users available when assigning roles on collections, datasets, and files.

How Can You Tell When Content is Locally FAIR?

The Dataverse UI adds a "Locally FAIR" tag to all collections, datasets, and files who's visibility is limited by the locally FAIR mechanism.

Why is Locally FAIR Support "Experimental"

The word "experimental" is used when functionality is new, may evolve signifcantly in future releases, and generally may require more effort to configure and manage and/or more effort to support than more mature functionality.

With the current Locally FAIR implementation, managers need to be aware that they are responsible for choosing collection settings compatible with Locally FAIR content, i.e. not using DOIs (whose metadata is publicly accessible) or publicly visible stores, etc. Users and managers should also be aware that some functionality that might expose Locally FAIR content, e.g. linking, may not be prohibited programmatically but should still be avoided. Similarly, users should be aware that functionality such as metrics and quotas may expose the existence of Locally FAIR content. If your Dataverse instance supports Locally FAIR data, you are encouraged to be an active participant in reporting any issues and suggesting further improvements.

Things to Keep in Mind

If your repository supports Locally FAIR content:

- published does not always mean public;
- search and browse results may vary depending on who is logged in;
- colleagues outside your authorized group may not be able to see the same datasets you can see;
- you should not share Locally FAIR content with others who don't have access themselves; and
- this functionality is experimental.

Enabling Locally FAIR Support

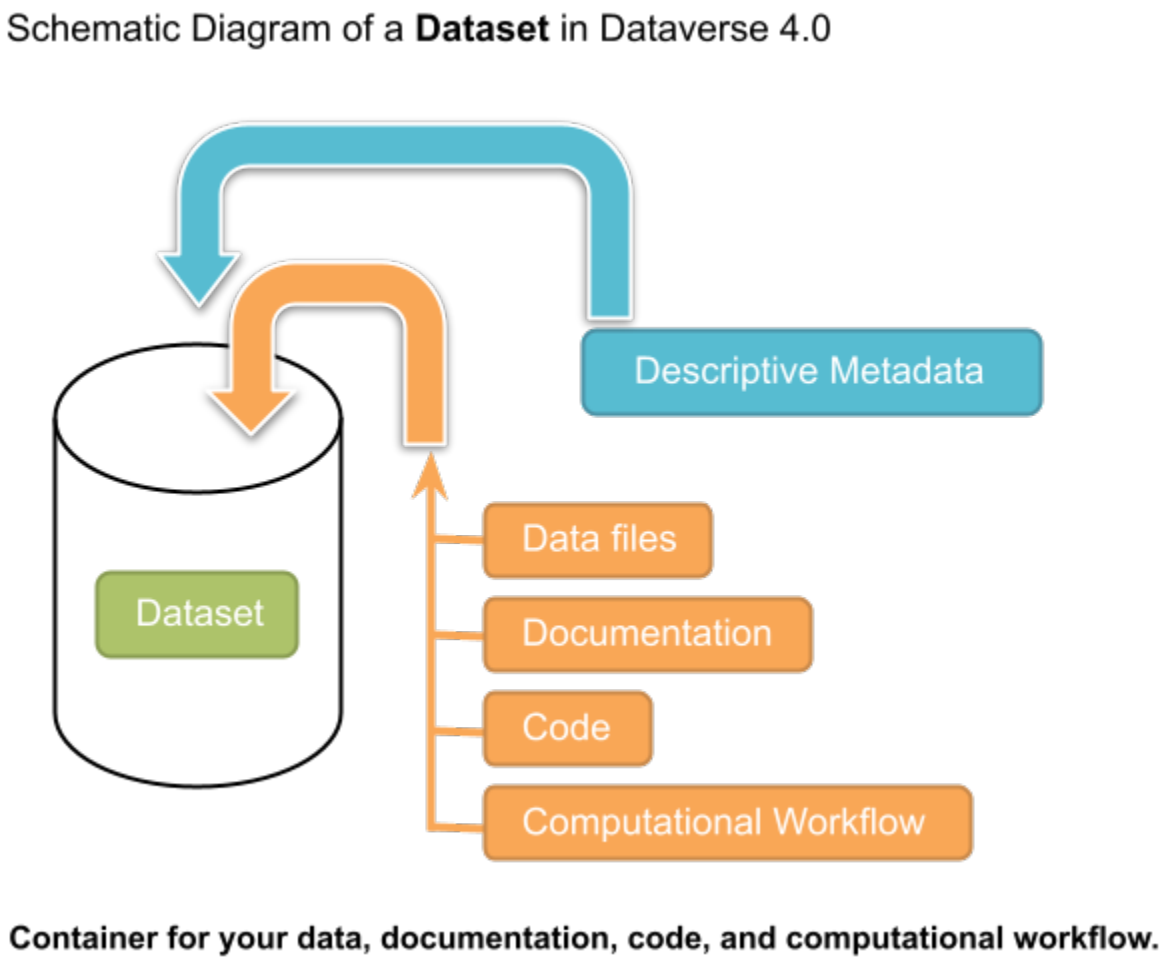
See *dataverse.feature.allow-locally-fair-data* in the Installation Guide.

Managing Locally FAIR Via API

See *List Locally FAIR Role Assignees for a Dataverse Collection* and related API enpoints in the API Guide.

2.4 Dataset + File Management

A dataset in a Dataverse installation is a container for your data, documentation, code, and the metadata describing this Dataset.



Contents:

- *Supported Metadata*
  - *Supported Metadata Export Formats*

- *Adding a New Dataset*
  - *Supported HTML Tags*
- *File Upload*
  - *HTTP Upload*
  - *Dropbox Upload*
  - *Folder Upload*
  - *Command-line DVUploader*
    - *Usage*
  - *Integrations Dashboard Uploader*
  - *Duplicate Files*
  - *BagIt Support*
- *File Handling*
  - *File Previews*
  - *Tabular Data Files*
  - *Research Code*
  - *Computational Workflow*
    - *Computational Workflow Definition*
    - *FAIR Computational Workflow*
    - *How to Create a Computational Workflow*
    - *How to Upload Your Computational Workflow*
    - *How to Describe Your Computational Workflow*
    - *How to Search for Computational Workflows*
  - *Astronomy (FITS)*
  - *GeoJSON*
  - *GeoTIFF*
  - *Shapefile*
  - *NetCDF and HDF5*
    - *H5Web Previewer*
    - *NcML*
    - *Geospatial Bounding Box*
  - *Compressed Files*
  - *Other File Types*
- *Restricted Files*
- *Edit Files*
  - *Edit File Metadata*

    - *Edit File Variable Metadata*
    - *File Path*
    - *File Tags*
- *Replace Files*
- *Terms*
    - *Choosing a License*
        * *License Selection and Professional Norms*
    - *Custom Terms of Use for Datasets*
    - *Restricted Files + Terms of Access*
    - *Creating and Depositing Differentially Private Metadata (Experimental)*
    - *Guestbook*
- *Roles & Permissions*
    - *Dataset-Level*
    - *File-Level*
- *Data Provenance*
    - *Dataset-Level*
    - *File-Level*
- *Thumbnails + Widgets*
    - *Thumbnails*
    - *Widgets*
        * *Dataset Widget*
        * *Dataset Citation Widget*
        * *Adding Widgets to an OpenScholar Website*
- *Publish Dataset*
- *Submit for Review*
- *Preview URL to Review Unpublished Dataset*
- *Embargoes*
- *Retention Periods*
- *Dataset Versions*
    - *Version Details*
- *Dataset Metrics and Make Data Count*
- *Cloud Storage + Computing*
    - *Cloud Computing*
    - *Cloud Storage Access*
- *Dataset Deaccession*

- *Dataset Types*
  - *Review Datasets*
    - *Review Dataset Overview*
    - *Creating a Review Dataset*
  - *Dataset Types and DataCite*

2.4.1 Supported Metadata

A dataset contains three levels of metadata:

1. **Citation Metadata**: any metadata that would be needed for generating a data citation and other general metadata that could be applied to any dataset;
2. **Domain Specific Metadata**: with specific support currently for Social Science, Life Science, Geospatial, and Astronomy datasets; and
3. **File-level Metadata**: varies depending on the type of data file - for more details see *File Handling* section below).

For more details about what Citation and Domain Specific Metadata is supported please see our *Appendix*.

Supported Metadata Export Formats

Once a dataset has been published, its metadata can be exported in a variety of other metadata standards and formats, which help make datasets more *discoverable* and usable in other systems, such as other data repositories. On each dataset page's metadata tab, the following exports are available:

- Croissant (example: `max-croissant.json`)
- Dublin Core
- DDI (Data Documentation Initiative Codebook 2.5)
- DDI HTML Codebook (A more human-readable, HTML version of the DDI Codebook 2.5 metadata export)
- DataCite 4
- JSON (native Dataverse Software format)
- OAI_ORE
- OpenAIRE
- Schema.org JSON-LD

Additional formats can be enabled. See *Inventory of External Exporters* in the Installation Guide. For example:

- RO-Crate: See also https://www.researchobject.org/ro-crate/dataverse

Each of these metadata exports contains the metadata of the most recently published version of the dataset.

For each dataset, links to each enabled metadata format are available programmatically via Signposting. For details, see *Signposting* in the Admin Guide and *Retrieve Signposting Information* in the API Guide.

2.4.2 Adding a New Dataset

1. Navigate to the Dataverse collection in which you want to add a dataset.
2. Click on the "Add Data" button and select "New Dataset" in the dropdown menu. **Note:** If you are on the root Dataverse collection, your My Data page or click the "Add Data" link in the navbar, the dataset you create will be hosted in the root Dataverse collection. You can change this by selecting another Dataverse collection you have proper permissions to create datasets in, from the Host Dataverse collection dropdown in the create dataset form. This option to choose will not be available after you create the dataset.
3. To quickly get started, enter at minimum all the required fields with an asterisk (e.g., the Dataset Title, Author Name, Description Text, Point of Contact Email, and Subject) to get a Data Citation with a DOI.
    1. When entering author identifiers, select the type from the dropdown (e.g. "ORCID") and under "Identifier" enter the full URL (e.g. "https://orcid.org/0000-0002-1825-0097") for identifiers that have a URL form. The shorter form of the unique identifier (e.g. "0000-0002-1825-0097") can also be entered, but URL form is preferred when available.
4. Scroll down to the "Files" section and click on "Select Files to Add" to add all the relevant files to your Dataset. You can also upload your files directly from your Dropbox. **Tip:** You can drag and drop or select multiple files at a time from your desktop directly into the upload widget. Your files will appear below the "Select Files to Add" button where you can add a description and tags (via the "Edit Tag" button) for each file. Additionally, an MD5 checksum will be added for each file. If you upload a tabular file a *Universal Numerical Fingerprint (UNF)* will be added to this file.
5. Click the "Save Dataset" button when you are done. Your unpublished dataset is now created.

Note: You can add additional metadata once you have completed the initial dataset creation by going to clicking the Edit button and selecting Metadata from the dropdown menu.

Supported HTML Tags

We currently only support the following HTML tags for any of our textbox metadata fields (i.e., Description) : <a>, <b>, <blockquote>, <br>, <code>, <del>, <dd>, <dl>, <dt>, , <hr>, <h1>-<h3>, <i>, <img>, <kbd>, <li>, <ol>, <p>, <pre>, <s>, <sup>, <sub>, , <strike>, <u>, <ul>.

2.4.3 File Upload

The Dataverse Software offers multiple methods of uploading files to a dataset. These upload methods are configurable by the administrator of a Dataverse installation, so you might not see some of these options on the Dataverse installation you're using.

If there are multiple upload options available, then you must choose which one to use for your dataset. A dataset may only use one upload method. Once you upload a file using one of the available upload methods, that method is locked in for that dataset. If you need to switch upload methods for a dataset that already contains files, then please contact Support by clicking on the Support link at the top of the application.

You can upload files to a dataset while first creating that dataset. You can also upload files after creating a dataset by clicking the "Edit" button at the top of the dataset page and from the dropdown list selecting "Files (Upload)" or clicking the "Upload Files" button above the files table in the Files tab. From either option you will be brought to the Upload Files page for that dataset.

Certain file types in a Dataverse installation are supported by additional functionality, which can include downloading in different formats, previews, file-level metadata preservation, file-level data citation with UNFs, and exploration through data visualization and analysis. See the *File Handling* section of this page for more information.

HTTP Upload

HTTP Upload is a common browser-based file upload tool you may be familiar with from other web applications. You can upload files via HTTP by selecting them from your browser or dragging and dropping them into the upload widget.

Once you have uploaded files, you will be able to edit file metadata, restrict access to files[1] , and/or add tags. Click "Save Changes" to complete the upload. If you uploaded a file by mistake, you can delete it before saving by clicking the checkbox to select the file, and then clicking the "Delete" button above the Files Table.

File upload limit size varies based on Dataverse installation. The file upload size limit can be found in the text above the HTTP upload widget. If you need to upload a very large file or a very large *number* of files, consider using DVUploader (see *Command-line DVUploader*).

Dropbox Upload

Some Dataverse installations support the ability to upload files directly from Dropbox. To do so, click the "Upload from Dropbox" button, log in to Dropbox in the pop-up window, and select the files you'd like to transfer over.

Folder Upload

Some Dataverse installations support the ability to upload files from a local folder and subfolders. To do this, click the "Upload from Folder" button, select the folder you wish to upload, select/unselect specific files, and click "Start Uploads". More detailed instructions are available in the DVWebloader wiki.

Command-line DVUploader

The open-source DVUploader tool is a stand-alone command-line Java application that uses the Dataverse installation's API to upload files to a specified Dataset. Since it can be installed by users, and requires no server-side configuration, it can be used with any Dataverse installation. It is intended as an alternative to uploading files through the Dataverse installation's web interface in situations where the web interface is inconvenient due to the number of files or file locations (spread across multiple directories, mixed with files that have already been uploaded or file types that should be excluded) or the need to automate uploads. Since it uses the Dataverse installation's API, transfers are limited in the same ways as HTTP uploads through the Dataverse installation's web interface in terms of size and performance. The DVUploader logs its activity and can be killed and restarted as desired. If stopped and resumed, it will continue processing from where it left off.

Usage

The DVUploader is open source and is available as source, as a Java jar, and with documentation at https://github.com/GlobalDataverseCommunityConsortium/dataverse-uploader. The DVUploader requires Java 1.8+. Users will need to install Java if they don't already have it and then download the latest release of the DVUploader - jar file. Users will need to know the URL of the Dataverse installation, the DOI of their existing dataset, and have generated an API Key for the Dataverse installation (an option in the user's profile menu).

Basic usage is to run the command:

```
java -jar DVUploader-*.jar -server=<Dataverse Installation URL> -did=<Dataset DOI> -key=
↪<User's API Key> <file or directory list>
```

[1] Some Dataverse installations do not allow this feature.

Additional command line arguments are available to make the DVUploader list what it would do without uploading, limit the number of files it uploads, recurse through sub-directories, verify fixity, exclude files with specific extensions or name patterns, and/or wait longer than 60 seconds for any Dataverse installation ingest lock to clear (e.g. while the previously uploaded file is processed, as discussed in the *File Handling* section below).

DVUploader is a community-developed tool, and its creation was primarily supported by the Texas Digital Library. Further information and support for DVUploader can be sought at the project's GitHub repository .

Integrations Dashboard Uploader

There is an experimental uploader described at *Integrations Dashboard* that provides a graphical user interface (GUI) for uploading files from a local file system and various remote locations such as GitHub.

Duplicate Files

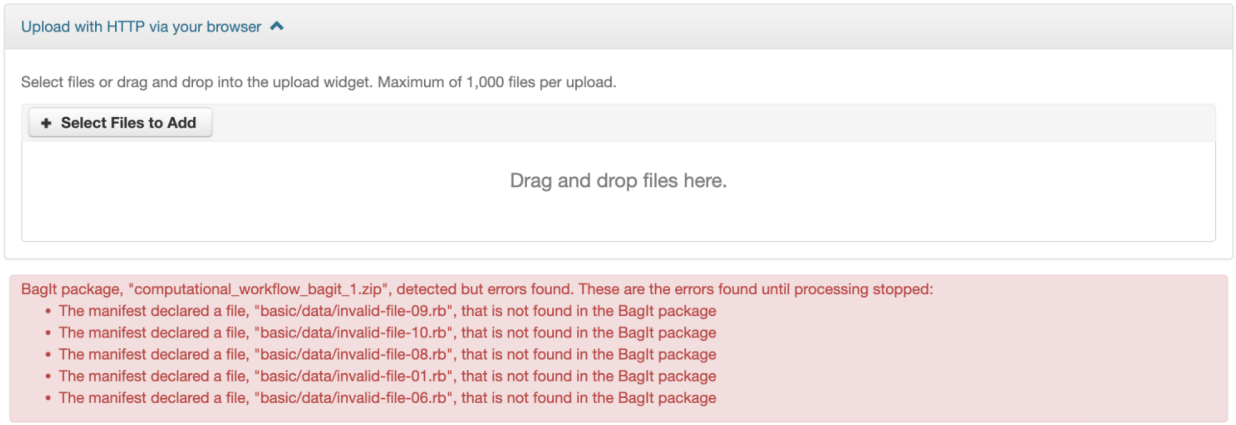
Beginning with Dataverse Software 5.0, the way a Dataverse installation handles duplicate files (filename and checksums) is changing to be more flexible. Specifically:

- Files with the same checksum can be included in a dataset, even if the files are in the same directory.
- Files with the same filename can be included in a dataset as long as the files are in different directories.
- If a user uploads a file to a directory where a file already exists with that directory/filename combination, the Dataverse installation will adjust the file path and names by adding "-1" or "-2" as applicable. This change will be visible in the list of files being uploaded.
- If the directory or name of an existing or newly uploaded file is edited in such a way that would create a directory/filename combination that already exists, the Dataverse installation will display an error.
- If a user attempts to replace a file with another file that has the same checksum, an error message will be displayed and the file will not be able to be replaced.
- If a user attempts to replace a file with a file that has the same checksum as a different file in the dataset, a warning will be displayed.

BagIt Support

BagIt is a set of hierarchical file system conventions designed to support disk-based storage and network transfer of arbitrary digital content. It offers several benefits such as integration with digital libraries, easy implementation, and transfer validation. See the Wikipedia article for more information.

If the Dataverse installation you are using has enabled BagIt file handling, when uploading BagIt files the repository will validate the checksum values listed in each BagIt's manifest file against the uploaded files and generate errors about any mismatches. The repository will identify a certain number of errors, such as the first five errors in each BagIt file, before reporting the errors.



You can fix the errors and reupload the BagIt files.

More information on how your admin can enable and configure the BagIt file handler can be found in the *Installation Guide*.

2.4.4 File Handling

Certain file types in the Dataverse installation are supported by additional functionality, which can include downloading in different formats, previews, file-level metadata preservation, file-level data citation; and exploration through data visualization and analysis. See the sections below for information about special functionality for specific file types.

File Previews

Dataverse installations can add previewers for common file types uploaded by their research communities. The previews appear on the file page. If a preview tool for a specific file type is available, the preview will be created and will display automatically, after terms have been agreed to or a guestbook entry has been made, if necessary. File previews are not available for restricted files unless they are being accessed using a Preview URL. See also *Preview URL to Review Unpublished Dataset*. When the dataset license is not the default license, users will be prompted to accept the license/data use agreement before the preview is shown. See also *Terms*.

Note

Some previewers run purely in the browser and make direct (JavaScript) requests back to the Dataverse API endpoints to retrieve file contents, metadata, or signed URLs. For these previewers to function when hosted on a different origin (e.g., a CDN or a separate previewer service), the Dataverse installation must have CORS enabled via *dataverse.cors.origin*. Administrators should configure the list of allowed origins to include the host serving the previewers.

Previewers are available for the following file types:

- Text
- PDF
- Markdown
- Tabular (CSV, Excel, etc., see *Tabular Data File Ingest*)
- Code (R, etc.)

- Images (PNG, GIF, JPG)
- Audio (MP3, MPEG, WAV, OGG, M4A)
- Video (MP4, OGG, Quicktime)
- Zip (preview and extract/download)
- HTML
- GeoJSON
- GeoTIFF
- Shapefile
- NetCDF/HDF5
- Hypothes.is

Additional file types will be added to the dataverse-previewers repo before they are listed above so please check there for the latest information or to request (or contribute!) an additional file previewer.

Installation of previewers is explained in the *External Tools* section of in the Admin Guide.

Tabular Data Files

Files in certain formats - Stata, SPSS, R, Excel (xlsx), CSV and TSV - may be ingested as tabular data (see *Tabular Data File Ingest* section of the User Guide for details). Tabular data files can be further explored and manipulated with *External Tools* if they have been enabled in the Dataverse installation you are using.

Additional download options available for tabular data (found in the same drop-down menu under the "Download" button):

- As tab-delimited data (with the variable names in the first row);
- The original file uploaded by the user;
- Saved as R data (if the original file was not in R format);
- Variable Metadata (as a DDI Codebook XML file);
- Data File Citation (currently in either RIS, EndNote XML, or BibTeX format).

Differentially Private (DP) Metadata can also be accessed for restricted tabular files if the data depositor has created a DP Metadata Release. See *Creating and Depositing Differentially Private Metadata (Experimental)* for more information.

Research Code

Code files - such as Stata, R, MATLAB, or Python files or scripts - have become a frequent addition to the research data deposited in Dataverse repositories. Research code is typically developed by few researchers with the primary goal of obtaining results, while its reproducibility and reuse aspects are sometimes overlooked. Because several independent studies reported issues trying to rerun research code, please consider the following guidelines if your dataset contains code.

The following are general guidelines applicable to all programming languages.

- Create a README text file in the top-level directory to introduce your project. It should answer questions that reviewers or reusers would likely have, such as how to install and use your code. If in doubt, consider using existing templates such as a README template for social science replication packages.
- Depending on the number of files in your dataset, consider having data and code in distinct directories, each of which should have some documentation like a README.

- Consider adding a license to your source code. You can do that by creating a LICENSE file in the dataset or by specifying the license(s) in the README or directly in the code. Find out more about code licenses at the Open Source Initiative webpage.
- If possible, use free and open-source file formats and software to make your research outputs more reusable and accessible.
- Consider testing your code in a clean environment before sharing it, as it could help you identify missing files or other errors. For example, your code should use relative file paths instead of absolute (or full) file paths, as they can cause an execution error.
- Consider providing notes (in the README) on the expected code outputs or adding tests in the code, which would ensure that its functionality is intact.

Capturing code dependencies will help other researchers recreate the necessary runtime environment. Without it, your code will not be able to run correctly (or at all). One option is to use platforms such as Jupyter Binder or Renku, which facilitate research reproducibility. For more information, have a look at *Integrations* in the Admin Guide, especially the sections on *Binder* and *Renku*. Another option is to use an automatic code dependency capture, which is often supported through the programming language. Here are a few examples:

- If you are using the conda package manager, you can export your environment with the command `conda env export > environment.yml`. For more information, see the official documentation.
- Python has multiple conventions for capturing its dependencies, but probably the best-known one is with the `requirements.txt` file, which is created using the command `pip freeze > requirements. txt`. Managing environments with `pip` is explained in the official documentation.
- If you are using the R programming language, create a file called `install.R`, and list all library dependencies that your code requires. This file should be executable in R to set up the environment. See also other strategies for capturing the environment proposed by RStudio in the official documentation.
- In case you are using multiple programming languages or different versions of the same language, consider using a containerization technology such as Docker. You can create a Dockerfile that builds your environment and deposit it within your dataset (see the official documentation). It is worth noting that creating a reliable Dockerfile may be tricky. If you choose this route, make sure to specify dependency versions and check out Docker's best practices.

Finally, automating your code can be immensely helpful to the code and research reviewers. Here are a few options on how to automate your code.

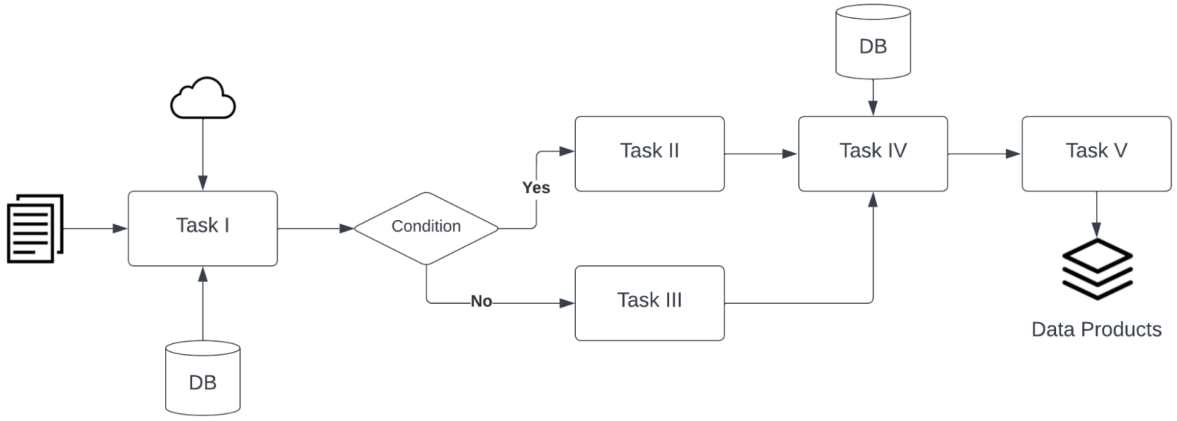
- A simple way to automate your code is using a bash script or Make. The Turing Way Community has a detailed guide on how to use the Make build automation tool.
- Consider using research workflow tools to automate your analysis. A popular workflow tool is called Common Workflow Language, and you can find more information about it from the Common Workflow Language User Guide.

Note: Capturing code dependencies and automating your code will create new files in your directory. Make sure to include them when depositing your dataset.

Computational Workflow

Computational Workflow Definition

Computational workflows precisely describe a multi-step process to coordinate multiple computational tasks and their data dependencies that lead to data products in a scientific application. The computational tasks take different forms, such as running code (e.g. Python, C++, MATLAB, R, Julia), invoking a service, calling a command-line tool, accessing a database (e.g. SQL, NoSQL), submitting a job to a compute cloud (e.g. on-premises cloud, AWS, GCP, Azure), and execution of data processing scripts or workflow. The following diagram shows an example of a computational workflow with multiple computational tasks.



FAIR Computational Workflow

The FAIR Principles (Findable, Accessible, Interoperable, Reusable) apply to computational workflows (https://doi.org/10.1162/dint_a_00033) in two areas: as FAIR data and as FAIR criteria for workflows as digital objects. In the FAIR data area, "*properly designed workflows contribute to FAIR data principles since they provide the metadata and provenance necessary to describe their data products, and they describe the involved data in a formalized, completely traceable way*" (https://doi.org/10.1162/dint_a_00033). Regarding the FAIR criteria for workflows as digital objects, "*workflows are research products in their own right, encapsulating methodological know-how that is to be found and published, accessed and cited, exchanged and combined with others, and reused as well as adapted*" (https://doi.org/10.1162/dint_a_00033).

How to Create a Computational Workflow

There are multiple approaches to creating computational workflows. You may consider standard frameworks and tools such as Common Workflow Language (CWL), Snakemake, Galaxy, Nextflow, Ruffus or *ad hoc* methods using different programming languages (e.g. Python, C++, MATLAB, Julia, R), notebooks (e.g. Jupyter Notebook, R Notebook, and MATLAB Live Script) and command-line interpreters (e.g. Bash). Each computational task is defined differently, but all meet the definition of a computational workflow and all result in data products. You can find a few examples of computational workflows in the following GitHub repositories, where each follows several aspects of FAIR principles:

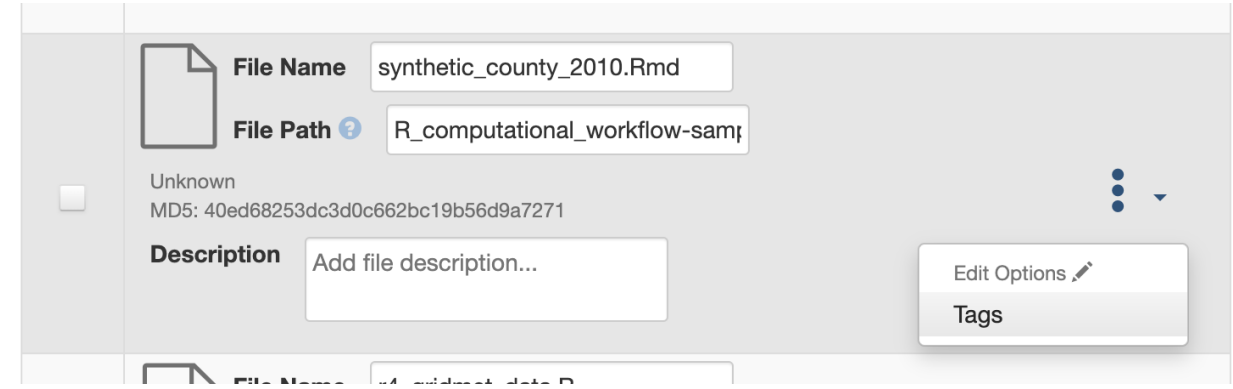
- Common Workflow Language (GitHub Repository URL)
- R Notebook (GitHub Repository URL)
- Jupyter Notebook (GitHub Repository URL)
- MATLAB Script (GitHub Repository URL)

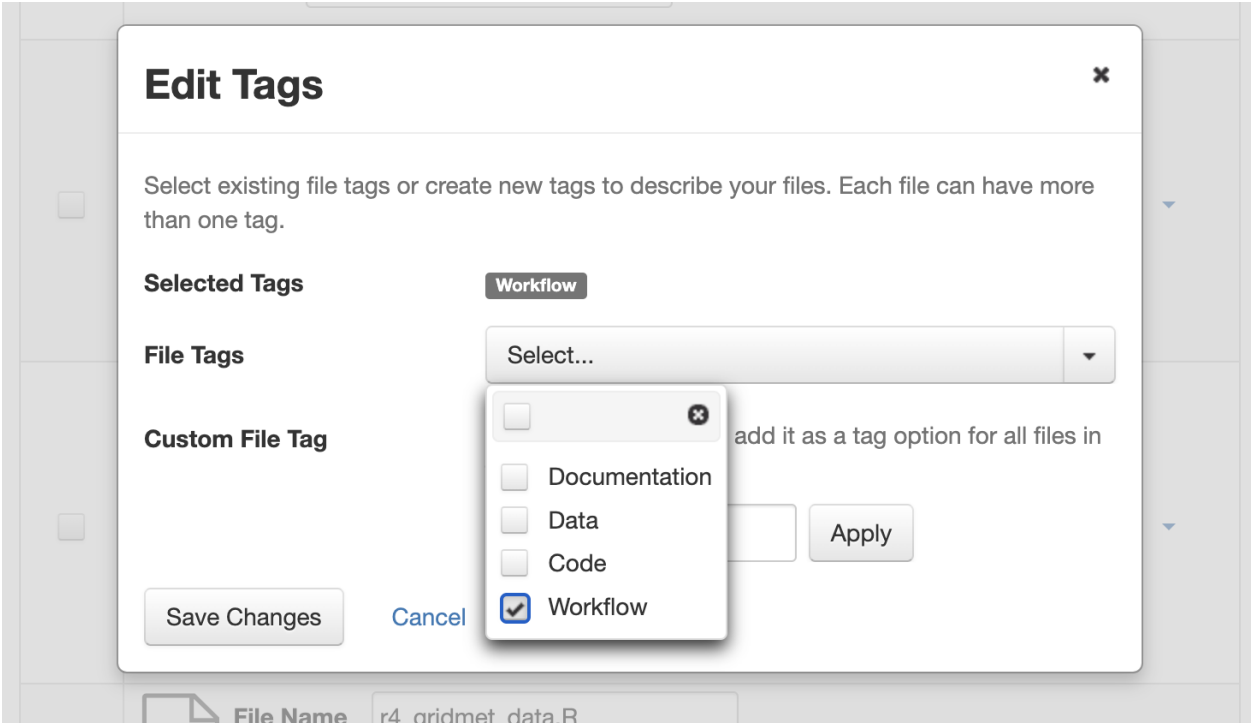
You are encouraged to review these examples when creating a computational workflow and publishing in a Dataverse repository.

At https://workflows.community, the Workflows Community Initiative offers resources for computational workflows, such as a list of workflow systems (https://workflows.community/systems) and other workflow registries (https://workflows.community/registries). The initiative also helps organize working groups related to workflows research, development and application.

How to Upload Your Computational Workflow

After you *upload your files*, you can apply a "Workflow" tag to your workflow files, such as your Snakemake or R Notebooks files, so that you and others can find them more easily among your deposit's other files.





How to Describe Your Computational Workflow

The Dataverse installation you are using may have enabled Computational Workflow metadata fields for your use. If so, when *editing your dataset metadata*, you will see the fields described below.

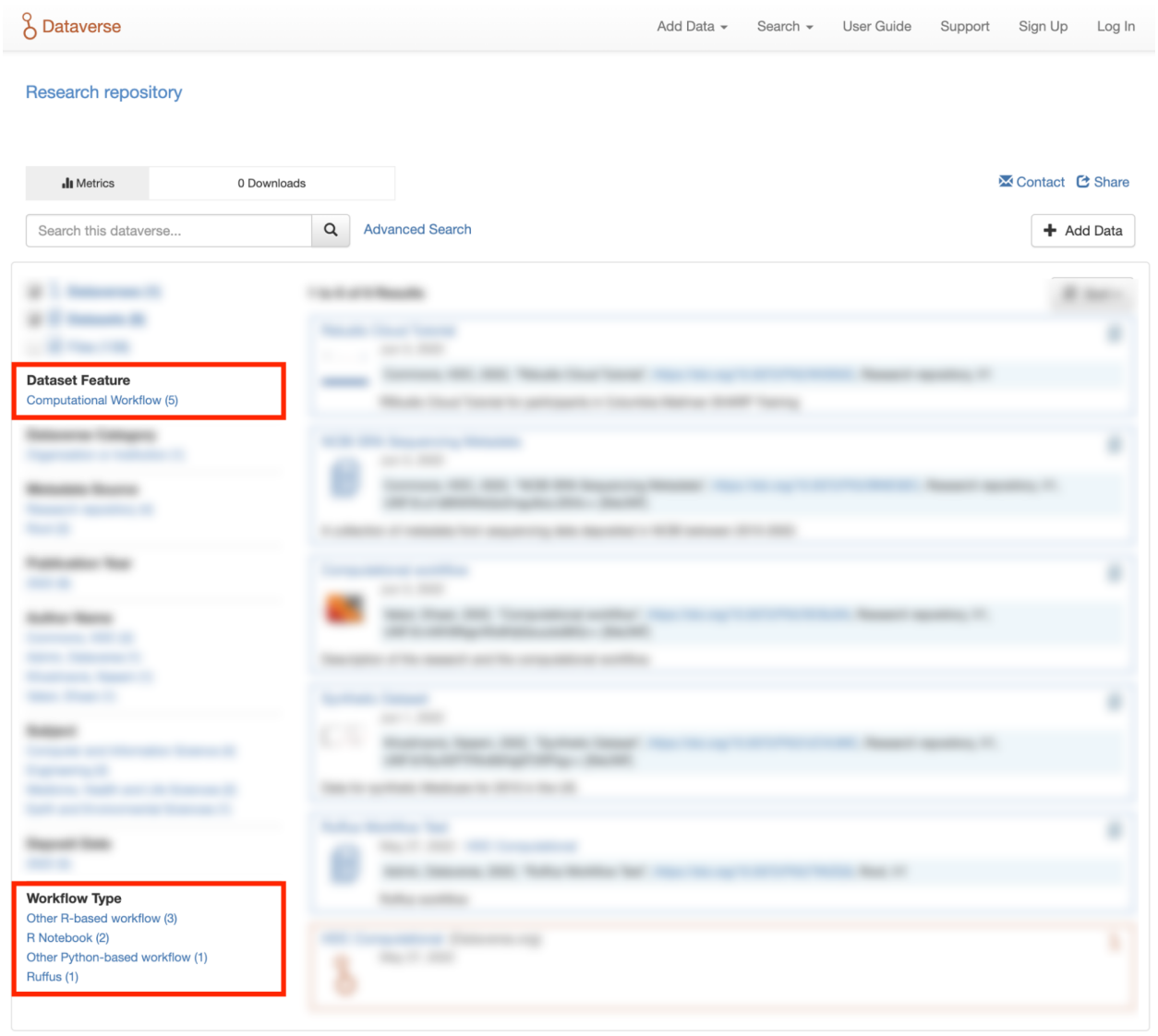
As described in the *Metadata References* section of the *Appendix*, the three fields are adapted from Bioschemas Computational Workflow Profile, version 1.0 and Codemeta:

- **Workflow Type**: The kind of Computational Workflow, which is designed to compose and execute a series of computational or data manipulation steps in a scientific application
- **External Code Repository URL**: A link to another public repository where the un-compiled, human-readable code and related code is also located (e.g., GitHub, GitLab, SVN)
- **Documentation**: A link (URL) to the documentation or text describing the Computational Workflow and its use

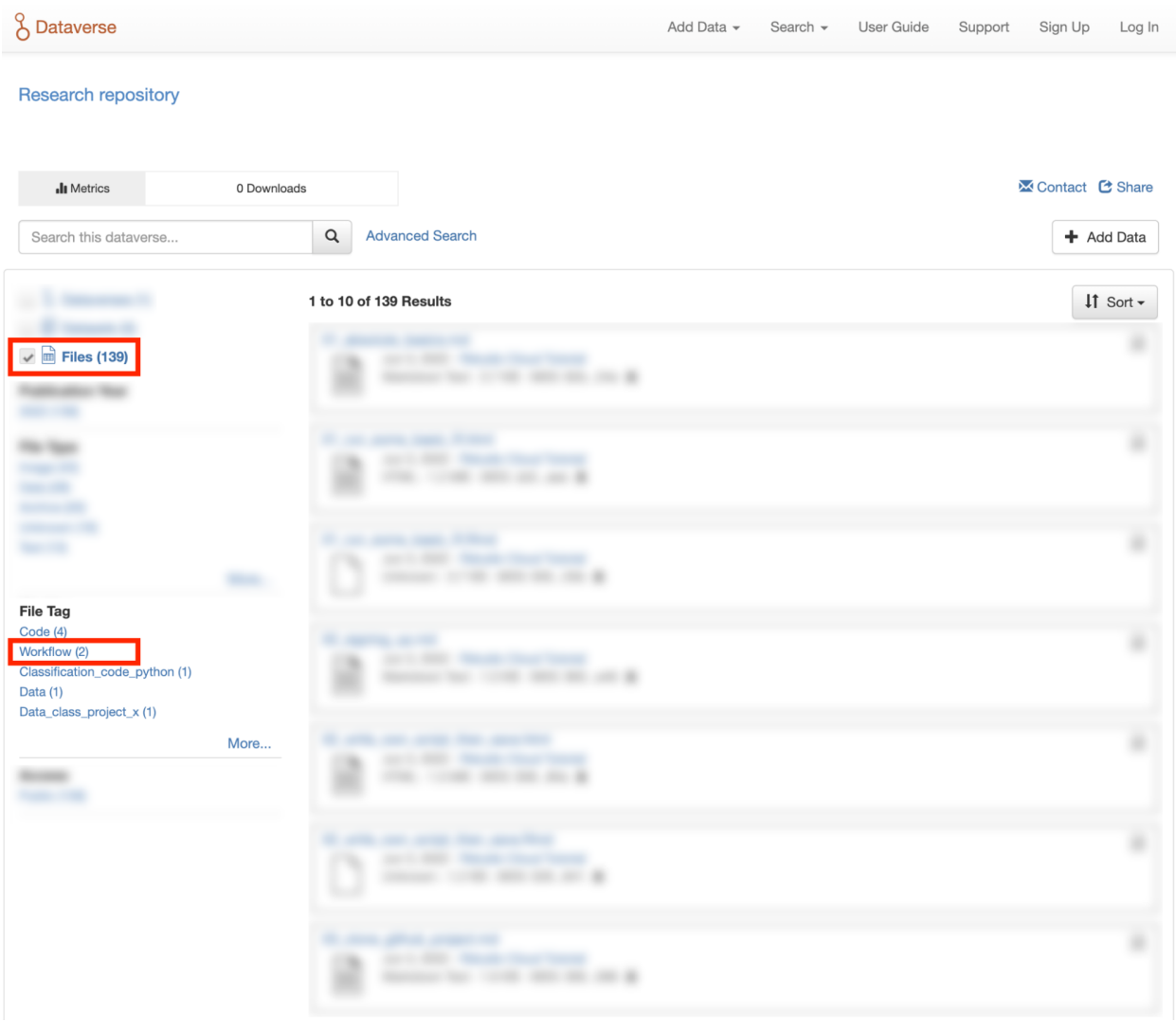
How to Search for Computational Workflows

If the search page of the Dataverse repository you are using includes a "Dataset Feature" facet with a Computational Workflows link, you can follow that link to find only datasets that contain computational workflows.

You can also search on the "Workflow Type" facet, if the Dataverse installation has the field enabled, to find datasets that contain certain types of computational workflows, such as workflows written in Common Workflow Language files or Jupyter Notebooks.



You can also search for files within datasets that have been tagged as "Workflow" files by clicking the Files checkbox to show only files and using the File Tag facet to show only files tagged as "Workflow".



Astronomy (FITS)

Metadata found in the header section of Flexible Image Transport System (FITS) files are automatically extracted by the Dataverse Software, aggregated and displayed in the Astronomy Domain-Specific Metadata of the Dataset that the file belongs to. This FITS file metadata, is therefore searchable and browsable (facets) at the Dataset-level.

GeoJSON

A map will be shown as a preview of GeoJSON files when the previewer has been enabled (see *File Previews*). See also a video demo of the GeoJSON previewer by its author, Kaitlin Newson.

GeoTIFF

A map is also displayed as a preview of GeoTiFF image files, whose previewer must be enabled (see *File Previews*). Since GeoTIFFs do not have their own mimetype, it is advisable to use this previewer only when GeoTIFFs are used (and not "normal" TIFs). For performance reasons, this previewer has a file size limit of 15 MB and a row/column limit of 50,000 so that larger files are not loaded.

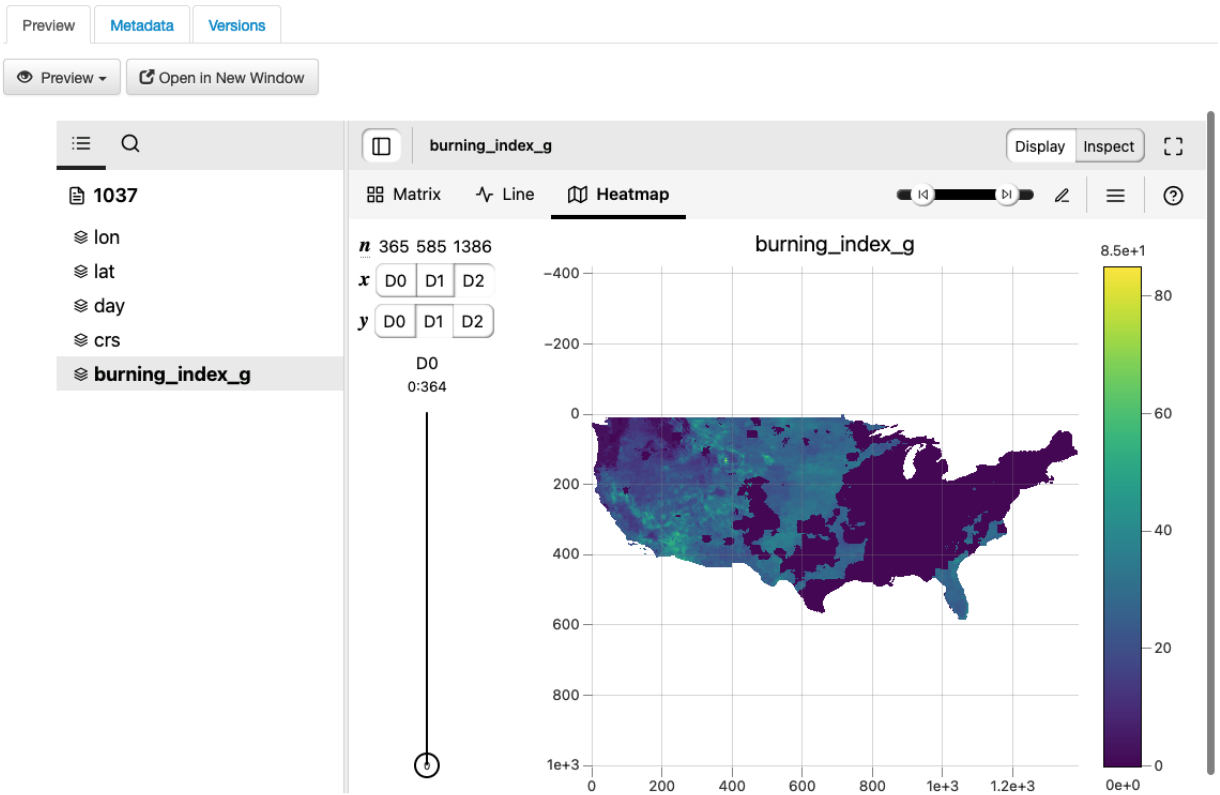
Shapefile

Another previewer can be enabled for shapefiles (see *File Previews*). This previewer only works with zipped shapefiles (see *Geospatial Data*). A file size limit of 20 MB is set for this previewer (also because of performance reasons).

NetCDF and HDF5

H5Web Previewer

NetCDF and HDF5 files can be explored and visualized with H5Web, which has been adapted into a file previewer tool (see *File Previews*) that can be enabled in your Dataverse installation.



NcML

For NetCDF and HDF5 files, an attempt will be made to extract metadata in NcML (XML) format and save it as a public auxiliary file, even if the file is restricted. (See also *Auxiliary File Support* in the Developer Guide.) A previewer for these NcML files is available (see *File Previews*).

Please note that only modern versions of these formats, the ones based on HDF5 such as NetCDF 4+ and HDF5 itself (rather than HDF4), will yield an NcML auxiliary file.

Geospatial Bounding Box

An attempt will be made to extract a geospatial bounding box (west, south, east, north) from NetCDF and HDF5 files and then insert these values into the geospatial metadata block, if enabled.

This is the mapping that is used:

- geospatial_lon_min: West Longitude
- geospatial_lon_max: East Longitude
- geospatial_lat_max: North Latitude
- geospatial_lat_min: South Latitude

Please note the following rules regarding these fields:

- West Longitude and East Longitude are expected to be in the range of -180 and 180. (When using *Geospatial Search*, you should use this range for longitude.)
- If West Longitude and East Longitude are both over 180 (outside the expected -180:180 range), 360 will be subtracted to shift the values from the 0:360 range to the expected -180:180 range.
- If either West Longitude or East Longitude are less than zero but the other longitude is greater than 180 (which would imply an indeterminate domain, a lack of clarity of if the domain is -180:180 or 0:360), metadata will be not be extracted.
- If the bounding box was successfully populated, the subsequent removal of the NetCDF or HDF5 file from the dataset does not automatically remove the bounding box from the dataset metadata. You must remove the bounding box manually, if desired.
- This feature is disabled if S3 direct upload is enabled (see *Features that are Disabled if S3 Direct Upload is Enabled*) unless *dataverse.netcdf.geo-extract-s3-direct-upload* has been set to true.

If the bounding box was successfully populated, *Geospatial Search* should be able to find it.

Compressed Files

Depending on the configuration, compressed files in .zip format are unpacked automatically. If a .zip file is not unpacked, it will upload as is. If the number of files inside are more than a set limit (1,000 by default, configurable by the Administrator), you will get an error message and the .zip file will upload as is.

If the uploaded .zip file contains a folder structure, the Dataverse installation will keep track of this structure. A file's location within this folder structure is displayed in the file metadata as the File Path. When you download the contents of the dataset, this folder structure will be preserved and files will appear in their original locations.

These folder names are subject to strict validation rules. Only the following characters are allowed: the alphanumerics, '_', '-', '.' and ' ' (white space). When a zip archive is uploaded, the folder names are automatically sanitized, with any invalid characters replaced by the '.' character. Any sequences of dots are further replaced with a single dot.

For example, the folder name `data&info/code=@137` will be converted to `data.info/code.137`. When uploading through the Web UI, the user can change the values further on the edit form presented, before clicking the 'Save' button.

> **Note**
>
> If you upload multiple .zip files to one dataset, any subdirectories that are identical across multiple .zips will be merged together when the user downloads the full dataset.

If a .zip file is not unpacked and Zip Previewer is installed (see *File Previews*), it will be possible for users to view the contents of the zip file and to download individual files from within the .zip.

Other File Types

There are several advanced options available for certain file types.

- Image files: .jpg, .png, and .tif files are able to be selected as the default thumbnail for a dataset. The selected thumbnail will appear on the search result card for that dataset.
- SPSS files: SPSS files can be tagged with the language they were originally coded in. This is found by clicking on Advanced Options and selecting the language from the list provided.

2.4.5 Restricted Files

When you restrict a file it cannot be downloaded unless permission has been granted.

Differentially Private (DP) Metadata can be accessed for restricted tabular files if the data depositor has created a DP Metadata Release. See *Creating and Depositing Differentially Private Metadata (Experimental)* for more information.

See also *Restricted Files + Terms of Access* and *Roles & Permissions*.

2.4.6 Edit Files

Edit File Metadata

Go to the dataset you would like to edit, where you will see the listing of files. Select the files you would like to edit by using either the Select All checkbox or individually selecting files. Next, click the "Edit Files" button above the file table and from the dropdown menu select if you would like to:

- Delete the selected files
- Edit the file metadata (file name, description) for the selected files
- Restrict the selected files
- Unrestrict the selected files (only if the selected files are restricted)
- Add tags to the selected files
- Embargo the selected files

You will not have to leave the dataset page to complete these action, except for editing file metadata, which will bring you to the Edit Files page. There you will have to click the "Save Changes" button to apply your edits and return to the dataset page.

If you restrict files, you will also prompted with a popup asking you to fill out the Terms of Access for the files. If Terms of Access already exist, you will be asked to confirm them. Note that some Dataverse installations do not allow for file restrictions.

Edit File Variable Metadata

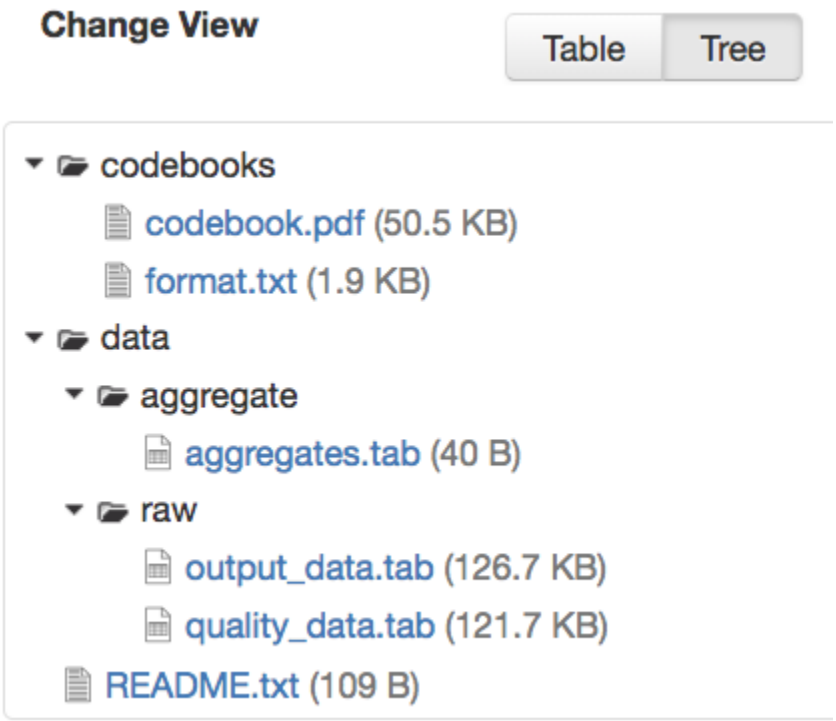
Variable Metadata can be edited directly through an API call (*API Guide: Editing Variable Level Metadata*) or by using the Dataverse Data Curation Tool.

File Path

The File Path metadata field is a Dataverse installation's way of representing a file's location in a folder structure. When a user uploads a .zip file containing a folder structure, the Dataverse installation automatically fills in the File Path information for each file contained in the .zip. If a user downloads the full dataset or a selection of files from it, they will receive a folder structure with each file positioned according to its File Path. Only one file with a given path and name may exist in a dataset. Editing a file to give it the same path and name as another file already existing in the dataset will cause an error.

A file's File Path can be manually added or edited on the Edit Files page. Changing a file's File Path will change its location in the folder structure that is created when a user downloads the full dataset or a selection of files from it.

If there is more than one file in the dataset, and once at least one of them has a non-empty directory path, the Dataset Page will present an option for switching between the traditional table view, and the tree-like view of the files showing the folder structure, as in the example below:



File Tags

File tags are comprised of custom, category (i.e. Documentation, Data, Code) and tabular data tags (i.e. Event, Genomics, Geospatial, Network, Panel, Survey, Time Series). Use the dropdown select menus as well as the custom file tag input to apply these tags to the selected files. There is also a Delete Tags feature that, if checked, will allow you to delete unused file tags within that dataset.

2.4.7 Replace Files

In cases where you would like to revise an existing file rather than add a new one, you can do so using our Replace File feature. This will allow you to track the history of this file across versions of your dataset, both before and after replacing it. This could be useful for updating your data or fixing mistakes in your data. Because replacing a file creates an explicit link between the previous dataset version and the current version, the file replace feature is not available for unpublished dataset drafts. Also note that replacing a file will not automatically carry over that file's metadata, but once the file is replaced then its original metadata can still be found by referencing the previous version of the file under the "Versions" tab of the file page.

To replace a file, go to the file page for that file, click on the "Edit" button, and from the dropdown list select "Replace". This will bring you to the Replace File page, where you can see the metadata for the most recently published version of the file and you can upload your replacement file. Once you have uploaded the replacement file, you can edit its name, description, and tags. When you're finished, click the "Save Changes" button.

After successfully replacing a file, a new dataset draft version will be created. A summary of your actions will be recorded in the "Versions" tab on on both the dataset page and file page. The Versions tab allows you to access all previous versions of the file across all previous versions of your dataset, including the old version of the file before you replaced it.

2.4.8 Terms

Dataset terms can be viewed and edited from the Terms tab of the dataset page, or under the Edit dropdown button of a Dataset. There, you can set up how users can use your data once they have downloaded it (via a standard license or, if allowed, custom terms), how they can access your data if you have files that are restricted (terms of access), and enable a Guestbook for your dataset so that you can track who is using your data and for what purposes. These are explained in further detail below:

Choosing a License

Each Dataverse installation provides a set of license(s) data can be released under, and whether users can specify custom terms instead (see below). One of the available licenses (often the Creative Commons CC0 Public Domain Dedication) serves as the default if you do not make an explicit choice. If you want to apply one of the other available licenses to your dataset, you can change it on the Terms tab of your Dataset page.

License Selection and Professional Norms

When selecting a license, data depositors should recognize that their data will be available internationally and, over the long term, may be used in new forms of research (for example, in machine learning where millions of datasets might be used in training). It is therefore useful to consider licenses that have been developed with awareness of international law and that place minimal restrictions on reuse.

For example, the Creative Commons organization defines a number of licenses that allow copyright holders to release their intellectual property more openly, with fewer legal restrictions than standard copyright enforces. (These licenses may or may not be available in the Dataverse instance you are using, but we expect them to be common in the community.) Each Creative Commons license typically specifies simple terms for how the IP must be used, reused, shared, and attributed and includes language intended to address variations in the laws of different countries.

In addition to these licenses, Creative Commons also provides the CC0 1.0 Universal (CC0 1.0) Public Domain Dedication which allows you to unambiguously waive all copyright control over your data in all jurisdictions worldwide. Data released with CC0 can be freely copied, modified, and distributed (even for commercial purposes) without violating copyright. In most parts of the world, factual data is exempt from copyright anyway, but applying CC0 removes all ambiguity and makes the legal copyright status of the data as clear as possible.

When available. CC0 can be a good choice for datasets because it facilitates reuse, extensibility, and long-term preservation of research data by assuring that the data can be safely handled by anyone without fear of potential copyright pitfalls.

Data depositors and data users should also understand that while licenses define legal use, they do not exempt a Dataverse installation's users from following ethical and professional norms in scholarly communications. For example, though CC0 waives a dataset owner's legal copyright controls over the data, users, as scholarly researchers, are still expected to cite the data they use, giving credit to the data's authors following ethical and professional norms in scholarly communications. This is true of other licenses as well - users should cite data as appropriate even if the specified license does not require it. The Dataverse Community Norms* detail additional areas where data users should follow societal norms and scientific best practices.

* **Legal Disclaimer:** these Community Norms are not a substitute for the CC0 waiver or custom terms and licenses applicable to each dataset. The Community Norms are not a binding contractual agreement, and downloading datasets from a Dataverse installation does not create a legal obligation to follow these policies.

Custom Terms of Use for Datasets

If the Dataverse you are using allows it, you may specify your own Custom Dataset Terms. To do so, select Custom Dataset Terms for your license, and a panel will appear allowing you to enter custom Terms of Use. You can also enter information in additional fields including Special Permissions, Restrictions, and Citation Requirements to further clarify how your Dataset may be accessed and used.

Here is an example of a Data Usage Agreement for datasets that have de-identified human subject data.

Restricted Files + Terms of Access

If you restrict any files in your dataset, you will be prompted by a pop-up to enter Terms of Access for the data. This can also be edited in the Terms tab or selecting Terms in the "Edit" dropdown button in the dataset. You may also allow users to request access for your restricted files by enabling "Request Access". To add more information about the Terms of Access, we have provided fields like Data Access Place, Availability Status, Contact for Access, etc. If you restrict a file, it will not have a preview shown on the file page.

Note: Some Dataverse installations do not allow for file restriction.

See also *Restricted Files*.

Creating and Depositing Differentially Private Metadata (Experimental)

Through an integration with tools from the OpenDP Project (opendp.org), the Dataverse Software offers an experimental workflow that allows a data depositor to create and deposit Differentially Private (DP) Metadata files, which can then be used for exploratory data analysis. This workflow allows researchers to view the DP metadata for a tabular file, determine whether or not the file contains useful information, and then make an informed decision about whether or not to request access to the original file.

If this integration has been enabled in your Dataverse installation, you can follow these steps to create a DP Metadata Release and make it available to researchers, while still keeping the files themselves restricted and able to be accessed after a successful access request.

- Deposit a tabular file and let the ingest process complete
- Restrict the File
- In the kebab next to the file on the dataset page, or from the "Edit Files" dropdown on the file page, click "OpenDP Tool"
- Go through the process to create a DP Metadata Release in the OpenDP tool, and at the end of the process deposit the DP Metadata Release back to the Dataverse installation
- Publish the Dataset

Once the dataset is published, users will be able to request access using the normal process, but will also have the option to download DP Statistics in order to get more information about the file.

Guestbook

This is where you will enable a particular Guestbook for your dataset, which is setup at the Dataverse collection level. For specific instructions please visit the *Dataset Guestbooks* section of the Dataverse Collection Management page.

2.4.9 Roles & Permissions

Dataverse installation user accounts can be granted roles that define which actions they are allowed to take on specific Dataverse collections, datasets, and/or files. Each role comes with a set of permissions, which define the specific actions that users may take. It is not possible to grant a role that comes with a permission that the granting user themselves does not have.

Roles and permissions may also be granted to groups. Groups can be defined as a set of Dataverse user accounts, a collection of IP addresses (e.g. all users of a library's computers), or a collection of all users who log in using a particular institutional login (e.g. everyone who logs in with a particular university's account credentials).

Dataset-Level

Admins or curators of a dataset can assign roles and permissions to the users of that dataset. If you are an admin or curator of a dataset, then you can get to the dataset permissions page by clicking the "Edit" button, highlighting "Permissions" from the dropdown list, and clicking "Dataset".

When you access a dataset's permissions page, you will see two sections:

Users/Groups: Here you can assign roles to specific users or groups, determining which actions they are permitted to take on your dataset. You can also reference a list of all users who have roles assigned to them for your dataset and remove their roles if you please. Some of the users listed may have roles assigned at the Dataverse collection level, in which case those roles can only be removed from the Dataverse collection permissions page.

Roles: Here you can reference a full list of roles that can be assigned to users of your dataset. Each role lists the permissions that it offers.

File-Level

If specific files in your dataset are restricted access, then you can grant specific users or groups access to those files while still keeping them restricted to the general public. If you are an admin or curator of a dataset, then you can get to the file-level permissions page by clicking the "Edit" button, highlighting "Permissions" from the dropdown list, and clicking "File".

When you access a dataset's file-level permissions page, you will see two sections:

Users/Groups: Here you can see which users or groups have been granted access to which files. You can click the "Grant Access to Users/Groups" button to see a box where you can grant access to specific files within your dataset to specific users or groups. If any users have requested access to a file in your dataset, you can grant or reject their access request here.

Restricted Files: In this section, you can see the same information, but broken down by each individual file in your dataset. For each file, you can click the "Assign Access" button to see a box where you can grant access to that file to specific users or groups.

2.4.10 Data Provenance

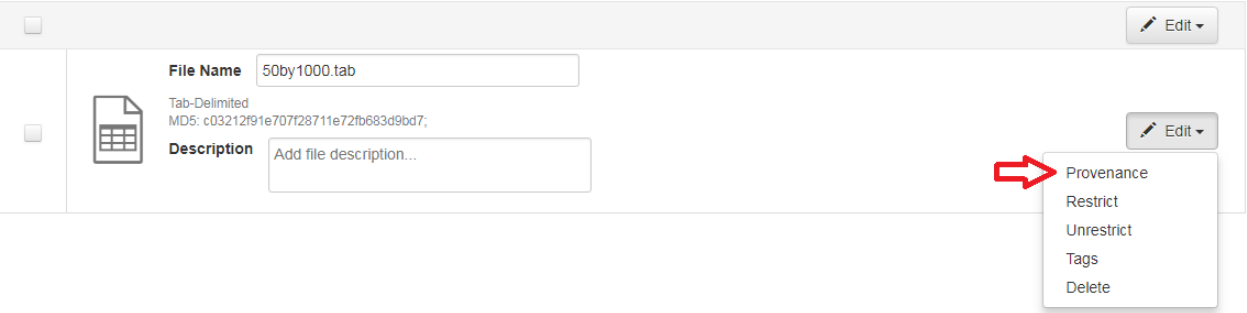
Dataset-Level

When configured, the Dataverse software can allow data depositors, curators, and administrators to provide information about why a new version of a dataset was created and/or how its contents differ from a prior version. These users can add an optional "Version Note" to a draft dataset version in the dataset page/versions tab or during publication. This information is publicly available via the user interface (dataset page/versions tab), API, and in metadata exports (including the DataCite, JSON, DDI, and OAI_ORE exports).

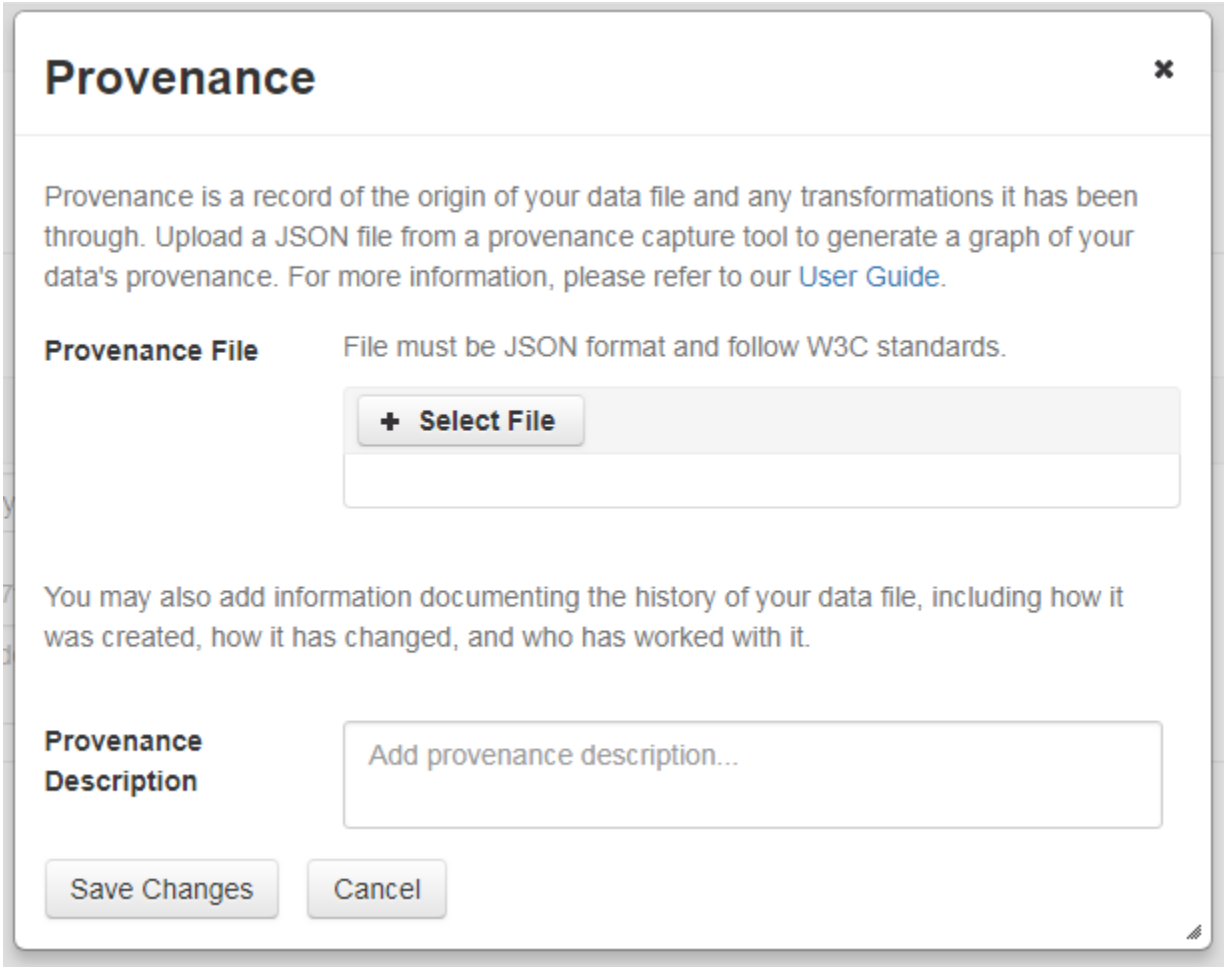
File-Level

Data Provenance is a record of where your data came from and how it reached its current form. It describes the origin of a data file, any transformations that have been made to that file, and any persons or organizations associated with that file. A data file's provenance can aid in reproducibility and compliance with legal regulations. When configured to support provenance, the Dataverse Software can help you keep track of your data's provenance. Currently, the Dataverse Software only makes provenance information available to those who have edit permissions on your dataset, but in the future we plan to expand this feature to make provenance information available to the public.

A Dataverse installation accepts provenance information in two forms: a *Provenance File* or a free-text *Provenance Description*. You can attach this provenance information to your data files in a Dataverse installation as part of the file upload process, by clicking Edit -> Provenance:



This will open a window where you can add your Provenance File and/or Provenance Description:



A **Provenance File** is the preferred way of submitting provenance information to a Dataverse installation because it provides a detailed and trustworthy record. Provenance files are typically generated during the process of data analysis, using provenance capture tools like provR, RDataTracker, NoWorkFlow, recordr, or CamFlow.

Once you upload a provenance file, the Dataverse installation will need some additional information in order to accurately connect it to your data file. Once provenance file upload finishes, an input box labeled "Connect entity" will appear under the file. Provenance files contain a list of "entities", which include your data file as well as any objects associated with it (e.g. a chart, a spellchecker, etc.). You will need to tell the Dataverse installation which entity within the provenance file represents your data file. You may type the name of the entity into the box, or click the arrow next to the box and select the entity from a list of all entities in the provenance file.

For more information on entities and the contents of provenance files, see the W3C PROV Model Primer. The uploaded file must be in JSON format as described in the PROV-JSON specification.

Once you've uploaded your Provenance File and connected the proper entity, you can hit the Preview button to view the raw JSON of the Provenance File. This can help you confirm that you've uploaded the right file. Be sure to double-check it, because the Provenance File will made *permanent* once it's finalized. At that point you will not be able to *replace*, *remove*, or otherwise *edit* the Provenance File. This ensures that the Provenance File maintains a stable, immutable record of the data file's history. This finalization of the Provenance File happens at different points depending on the status of your data file. If this is a brand new data file that has never been published before, then its associated Provenance File will be made permanent once you publish the dataset. If this data file *has* been published in a previous version of your dataset, then its associated Provenance File will be made permanent as soon as you upload the Provenance File and click "Save Changes" on the warning popup.

A **Provenance Description** allows you to add more provenance information in addition to or in place of a provenance file. This is a free-text field that allows you to enter any information you feel might be relevant to those interested in learning about the provenance of your data. This might be a good place to describe provenance factors like what operating system you used when working with the data file, what functions or libraries you used, how data was merged into the file, what version of the file you used, etc. The Provenance Description is not as useful or trustworthy as a provenance file, but it can still provide value. Unlike the Provenance File, the Provenance Description is never made permanent: you can always edit, remove, or replace it at any time.

You can return to attach provenance to your data file later on by clicking the "Add + Edit Metadata" button on the file page, and then clicking the "Edit -> Provenance" button.

2.4.11 Thumbnails + Widgets

Thumbnails

Thumbnail images can be assigned to a dataset manually or automatically. The thumbnail for a dataset appears on the search result card for that dataset and on the dataset page itself. If a dataset contains one or more data files that a Dataverse installation recognizes as an image, then one of those images is automatically selected as the dataset thumbnail.

If you would like to manually select your dataset's thumbnail, you can do so by clicking the "Edit" button on your dataset, and selecting "Thumbnails + Widgets" from the dropdown menu.

On this page, under the Thumbnail tab you will see three possible actions.

Select Available File: Click the "Select Thumbnail" button to choose an image from your dataset to use as the dataset thumbnail.

Upload New File: Upload an image file from your computer to use as the dataset thumbnail. While by default your thumbnail image is drawn from a file in your dataset, this will allow you to upload a separate image file to use as your dataset thumbnail. This uploaded image file will only be used as the dataset thumbnail; it will not be stored as a data file in your dataset.

Remove Thumbnail: If you click the "Remove" button under the thumbnail image, you will remove the dataset's current thumbnail. The Dataset will then revert to displaying a basic default icon as the dataset thumbnail.

When you're finished on this page, be sure to click "Save Changes" to save what you've done.

Note: If you prefer, it is also possible to set an image file in your dataset as your thumbnail by selecting the file, going to Edit Files -> Metadata, and using the "Set Thumbnail" button.

Widgets

The Widgets feature provides you with code for your personal website so your dataset can be displayed. There are two types of Widgets for a dataset: the Dataset Widget and the Dataset Citation Widget. Widgets are found by going to your dataset page, clicking the "Edit" button (the one with the pencil icon) and selecting "Thumbnails + Widgets" from the dropdown menu.

In the Widgets tab, you can copy and paste the code snippets for the widget you would like to add to your website. If you need to adjust the height of the widget on your website, you may do so by editing the *heightPx=500* parameter in the code snippet.

Dataset Widget

The Dataset Widget allows the citation, metadata, files and terms of your dataset to be displayed on your website. When someone downloads a data file in the widget, it will download directly from the datasets on your website. If a file is restricted, they will be directed to your Dataverse installation to log in, instead of logging in through the widget on your site.

To edit your dataset, you will need to return to the Dataverse installation where the dataset is stored. You can easily do this by clicking on the link that says "Data Stored in (Name) Dataverse Collection" found in the bottom of the widget.

Dataset Citation Widget

The Dataset Citation Widget will provide a citation for your dataset on your personal or project website. Users can download the citation in various formats by using the Cite Data button. The persistent URL in the citation will direct users to the dataset in your Dataverse installation.

Adding Widgets to an OpenScholar Website

1. Log in to your OpenScholar website
2. Either build a new page or navigate to the page you would like to use to show the Dataverse collection and dataset widgets.
3. Click on the Settings Cog and select Layout
4. At the top right, select Add New Widget and under Misc. you will see the Dataverse Collection and the Dataverse Dataset Citation Widgets. Click on the widget you would like to add, fill out the form, and then drag it to where you would like it to display in the page.

2.4.12 Publish Dataset

When you publish a dataset (available to an Admin, Curator, or any custom role which has this level of permission assigned), you make it available to the public so that other users can browse or search for it. Once your dataset is ready to go public, go to your dataset page and click on the "Publish" button on the right hand side of the page. A pop-up will appear to confirm that you are ready to actually Publish since once a dataset is made public it can no longer be unpublished.

Before the Dataverse installation finalizes the publication of the dataset, it will attempt to validate all the physical files in it, to make sure they are present and intact. In an unlikely event that any files fail the validation, you will see an error message informing you that the problem must be fixed by the local Dataverse Installation Admin before the dataset can be published.

Whenever you edit your dataset, you are able to publish a new version of the dataset. The publish dataset button will reappear whenever you edit the metadata of the dataset or add a file.

Note: Prior to publishing your dataset the Data Citation will indicate that this is a draft but the "DRAFT VERSION" text will be removed as soon as you Publish.

2.4.13 Submit for Review

If you have a Contributor role (can edit metadata, upload files, and edit files, edit Terms, Guestbook, and submit datasets for review) in a Dataverse collection you can submit your dataset for review when you have finished uploading your files and filling in all of the relevant metadata fields. To submit your dataset for review, go to your dataset and click the "Submit for Review" button, which is located next to the "Edit" button on the upper-right. In the confirmation popup, you can review your selection of license (or custom terms, if available). Once you have confirmed the submission, the Admin or Curator for this Dataverse collection will be notified to review this dataset before they decide to either publish the dataset or click "Return to Author". If the dataset is published, the contributor will be notified that it is now published. If the dataset is returned to the author, the contributor of this dataset will be notified that they need to make modifications before it can be submitted for review again.

2.4.14 Preview URL to Review Unpublished Dataset

Creating a Preview URL for a draft version of your dataset allows you to share your dataset (for viewing and downloading files, including *restricted* and *embargoed* files) before it is published to a wide group of people who might not have a user account on the Dataverse installation. Anyone you send the Preview URL to will not have to log in to the Dataverse installation to view the unpublished dataset. Once a dataset has been published, you may create new General Preview URLs for subsequent draft versions, but the Anonymous Preview URL will no longer be available.

Note: To create a Preview URL, you must have the *ManageDatasetPermissions* permission for your draft dataset, usually given by the *roles* *Curator* or *Administrator*. Take into account that before being able to create an Anonymous Preview URL, your system administrator much configure *a setting* that indicates which fields will be withheld (Author Name, for example) when dataset access occurs via a Preview Url with Anonymized Access.

1. Go to your unpublished dataset
2. Select the "Edit" button
3. Select "Preview URL" in the dropdown menu
4. In the pop-up select "Create General Preview URL" or "Create Anonymous Preview URL". The latter supports anonymous review by removing author names and other potentially identifying information from citations, version history tables, and some metadata fields (as configured by the administrator).
5. Copy the Preview URL which has been created for this dataset and it can now be shared with anyone you wish to have access to view or download files in your unpublished dataset.

To disable a Preview URL and to revoke access, follow the same steps as above until step #3 when you return to the popup, click the "Disable Preview URL" button.

Note: Before distributing an anonymized Preview URL it is recommended that you view the dataset as a potential user to verify that the metadata available does not reveal authorship, etc.

1. Create Anonymous Preview URL for your unpublished dataset via the Preview URL popup from Edit Dataset button
2. Copy the Anonymous Preview URL to your clipboard
3. Log out of Dataverse application
4. Open the dataset using the Anonymous Preview URL you plan to distribute to view it as a reviewer would.
5. It may be necessary for you to further edit your draft dataset's metadata to remove identifying items before you distribute the Anonymous Preview URL

Note that only one Preview URL (normal or with anonymized access) can be configured per dataset at a time.

2.4.15 Embargoes

A Dataverse instance may be configured to support file-level embargoes. Embargoes make file content inaccessible after a dataset version is published - until the embargo end date. A reason for the embargo may be supplied when creating the embargo. A reason may be *required* in some Dataverse instances. This means that file previews and the ability to download files will be blocked. The effect is similar to when a file is restricted except that the embargo will end at the specified date without further action and during the embargo, requests for file access cannot be made. Embargoes of files in a version 1.0 dataset may also affect the date shown in the dataset and file citations. The recommended practice is for the citation to reflect the date on which all embargoes on files in version 1.0 end. (Since Dataverse creates one persistent identifier per dataset and doesn't create new ones for each version, the publication of later versions, with or without embargoed files, does not affect the citation date.)

Embargoes are intended to support use cases where, for example, a journal or project team allows a period after publication of a dataset and/or the associated paper, during which the authors still have sole access to the data. Setting an embargo on relevant files and publishing the dataset in Dataverse publicizes the persistent identifier (e.g. DOI or Handle) for the dataset (and files if the instance is configured to create persistent identifiers for them) and makes the metadata, and any the content of un-embargoed files immediately available, but automatically denies access to any embargoed files until the specified embargoes expire. Once a dataset with embargoed files has been published, no further action is needed to cause the embargoed files to become accessible as of the specified embargo end date. (Note that embargoes can be set along with using the 'restrict' functionality on files. The restricted status will affect their availability as normal (and described elsewhere) once the embargo expires.)

- Setting the same embargo on all files in the dataset can be seen as providing a dataset-level embargo - making the dataset persistent identifier and metadata available but restricting access to all files.
- "Rolling" embargoes on time-series data can be supported by publishing multiple dataset versions and adding new embargoes on the files added in that version. For example, every year, files containing the prior year's results can be added to a dataset and given an embargo ending one year later than the embargoes set in the last dataset version, and the new dataset version can then be published. The datafiles published in the different versions will become available when their individual embargoes expire at yearly intervals.

As the primary use case of embargoes is to make the existence of data known now, with a promise (to a journal, project team, etc.) that the data itself will become available at a given future date, users cannot change an embargo once a dataset version is published. Dataverse instance administrators do have the ability to correct mistakes and make changes if/when circumstances warrant.

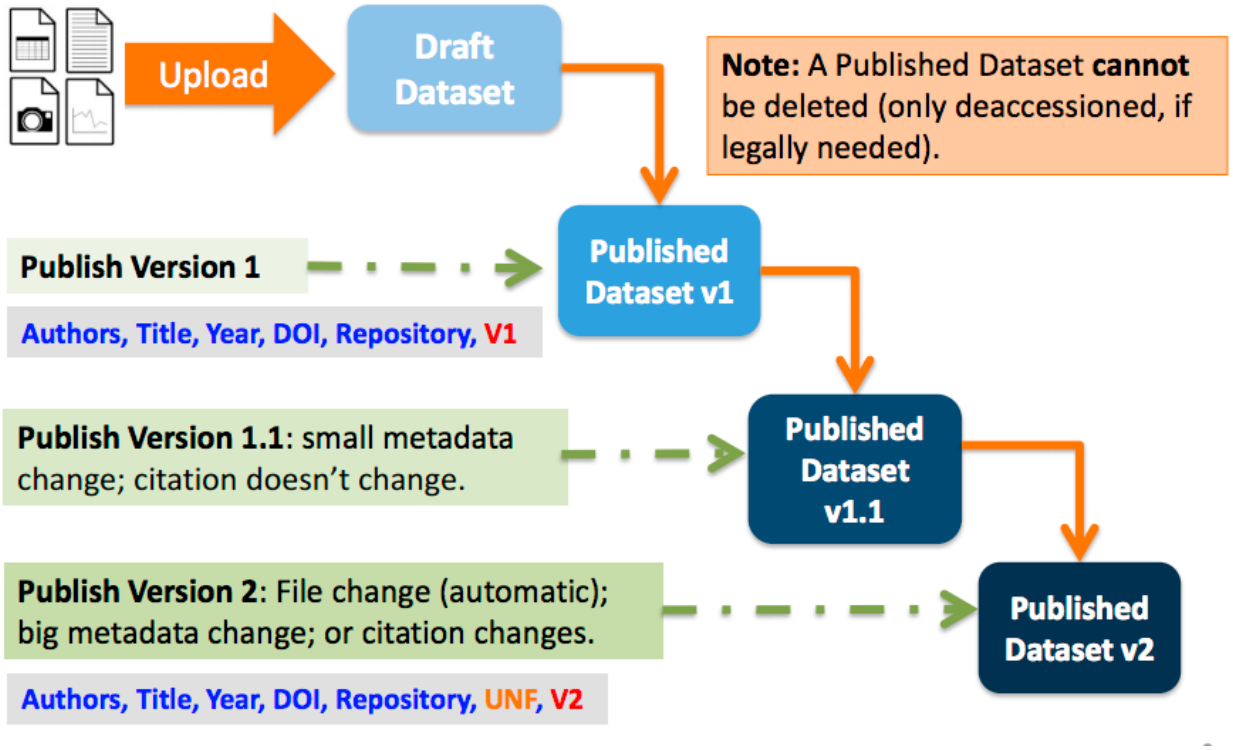
2.4.16 Retention Periods

Support for file-level retention periods can also be configured in a Dataverse instance. Retention periods make file content inaccessible after the retention period end date. This means that file previews and the ability to download files will be blocked. The effect is similar to when a file is restricted except that the retention periods will end at the specified date without further action and after the retention periods expires, requests for file access cannot be made.

Retention periods are intended to support use cases where files must be made unavailable - and in most cases destroyed, e.g. to meet legal requirements - after a certain period or date. Actual destruction is not automatically handled, but would have to be done on the storage if needed.

2.4.17 Dataset Versions

Versioning is important for long-term research data management where metadata and/or files are updated over time. It is used to track any metadata or file changes (e.g., by uploading a new file, changing file metadata, adding or editing metadata) once you have published your dataset.



Once you edit your published dataset, a draft version will be created. To publish this draft version, use the "Publish Dataset" button at the top right side of the page.

If files were added or removed, or if your dataset's previous version was deaccessioned, you must agree to publish the draft as a major version, such as version 2.0. Otherwise, you can choose to publish the draft as a major version or as a minor version, such as version 1.1.

On the Versions tab of a dataset page, there is a versions table that displays the version history of the dataset. You can use the version number links in this table to navigate between the different versions of the dataset, including the unpublished draft version, if you have permission to access it.

There is also a Versions tab on the file page. The versions table for a file displays the same information as the dataset, but the summaries are filtered down to only show the actions related to that file. If a new dataset version were created without any changes to an individual file, that file's version summary for that dataset version would read "No changes associated with this version".

Version Details

To view exactly what has changed, starting from the originally published version to any subsequent published versions: click the Versions tab on the dataset page to see all versions and changes made for that particular dataset.

Once you have more than one version (this can simply be version 1 and a draft), you can click the "View Details" link next to each summary to learn more about the metadata fields and files that were either added or edited. You can also click the checkboxes to select any two dataset versions, then click the "View Differences" button to open the Version Differences Details popup and compare the differences between them.

2.4.18 Dataset Metrics and Make Data Count

All Dataverse installations count file downloads. These file download counts are aggregated and reported at the Dataset level as well as at the file level.

Some Dataverse installations also have support for expanded metrics at the dataset level for views, file downloads, and citations using Make Data Count standards. Make Data Count is a project to collect and standardize metrics on data use, especially views, downloads, and citations. Citations for datasets are retrieved from Crossref via DataCite using Make Data Count standards.

For the specific API calls for Make Data Count, see *Dataset Metrics* in the *Native API* section of the API Guide.

2.4.19 Cloud Storage + Computing

Dataverse installations can be configured to facilitate cloud-based storage and/or computing (this feature is considered experimental at this time, and some of the kinks are still being worked out). While the default configuration for the Dataverse Software uses a local file system for storing data, a cloud-enabled Dataverse installation can use a Swift object storage database for its data. This allows users to perform computations on data using an integrated cloud computing environment.

Cloud Computing

The "Compute" button on dataset and file pages will allow you to compute on a single dataset, multiple datasets, or a single file. You can use it to build a compute batch and go directly to the cloud computing environment that is integrated with a Dataverse installation.

Cloud Storage Access

If you need to access a dataset in a more flexible way than the Compute button provides, then you can use the Cloud Storage Access box on the dataset page to copy the dataset's container name. This unique identifier can then be used to allow direct access to the dataset.

2.4.20 Dataset Deaccession

Warning

It is not recommended that you deaccession a dataset or a version of a dataset. This is a very serious action that should only occur if there is a legal or valid reason for the dataset to no longer be accessible to the public. If you absolutely must deaccession, you can deaccession a version of a dataset or an entire dataset.

To deaccession, go to your published dataset (or add a new one and publish it), click the "Edit" button, and from the dropdown menu select "Deaccession Dataset". If you have multiple versions of a dataset, you can select here which versions you want to deaccession or choose to deaccession the entire dataset.

You must also include a reason as to why this dataset was deaccessioned. Select the most appropriate reason from the dropdown list of options. If you select "Other", you must also provide additional information.

Add more information as to why this was deaccessioned in the free-text box. If the dataset has moved to a different repository or site you are encouraged to include a URL (preferably persistent) for users to continue to be able to access this dataset in the future.

If you deaccession the most recently published version of the dataset but not all versions of the dataset, you may then revisit an earlier version and create a new non-deaccessioned draft for the dataset. For example, imagine you have a version 1 and version 2 of a dataset, both published, and you deaccession version 2. You may then edit version 1 of the dataset and a new draft version will be created.

Important Note: A tombstone landing page with the basic citation metadata will always be accessible to the public if they use the persistent URL (Handle or DOI) provided in the citation for that dataset. Users will not be able to see any of the files or additional metadata that were previously available prior to deaccession.

2.4.21 Dataset Types

> **Note**
>
> Development of the dataset types feature is ongoing. Please see https://github.com/IQSS/dataverse-pm/issues/307 for details.

The vision for dataset types is to have variations on datasets. The best documented use case is *Review Datasets*, as explained below, but other types of datasets are possible such as software datasets (see *Add Dataset Type* for an example) or workflow datasets.

Out of the box, all datasets have a dataset type of "dataset", which is the traditional research data related dataset in Dataverse. Superusers can add additional types using the *Add Dataset Type* API endpoint. These additional dataset types cannot be used until a superuser has allowed them on a per-collection basis using the *Change Collection Attributes* API endpoint (by passing `allowedDatasetTypes`).

Dataset types can be listed, added, or deleted via API. See *Dataset Types* in the API Guide for more.

Currently, specifying a type for a dataset can only be done via API and only when the dataset is created. (The type can't be changed afterward.) For details, see the following sections of the API guide:

- *Create a Dataset with a Dataset Type (Software, Workflow, Review, etc.)* (Native API)
- *Create a Dataset with a Dataset Type* (Semantic API)
- *Import a Dataset with a Dataset Type (Software, Workflow, Review, etc.)*

Once more than one type appears in dataset search results, a facet called "Dataset Type" will appear allowing you to filter down to a certain type.

Dataset types can be linked with metadata blocks to make fields from those blocks available when datasets of that type are created or edited (via API). See *Link Dataset Type with Metadata Blocks* and *List Metadata Blocks Defined on a Dataverse Collection* for details.

Dataset types can change the "type" in Citation Style Language (CSL) output. See *Get Citation In Other Formats* for details.

If your installation is configured to use DataCite as a persistent ID (PID) provider, the dataset type may be sent to DataCite as `resourceTypeGeneral` (see also upstream schema). See the table under *Dataset Types and DataCite* for details.

Review Datasets

> **Note**
>
> This is an experimental feature.

Review Dataset Overview

Review datasets are a specialized type of dataset that can be used to review resources (such as datasets) in the Dataverse installation itself or resources in external data repositories.

This feature is only available via API and only if it has been configured by a superuser for your collection. See *Configure a Collection for Review Datasets* for details.

In the recommended setup, a collection is created that is managed by a research community, typically approved at the installation level.

In a typical use case, the reviews will be generated by these research communities based on the aggregation of scores for a particular domain by community-identified experts. These scores are stored in a custom metadata block, a rubric. An additional metadata block is required to hold information about the review itself, such a pointer to the resource being reviewed.

We recommend implementing a policy where there is only one review of a given resource per collection.

Almost all functionality is the same between regular datasets and review datasets. Review datasets build upon existing dataset functionality such as custom metadata blocks, versioning, publishing workflows, permissions, and file handling.

Review datasets build on the *Dataset Types* feature, allowing users to choose a dataset type of "review", which leads to a couple differences from regular datasets.

First, when multiple dataset types exist, a "Dataset Type" search facet appears that allows users to narrow results to the various kinds of dataset types that have been added, such as dataset, review, software, workflow, etc. (Under the "Collections, Datasets, Files" area, review datasets are considered datasets.)

Second, when review datasets are published, different `resourceType` metadata is sent to DataCite. Review datasets send "Other" for the field `resourceTypeGeneral` ("Work Type" in the UI at https://commons.datacite.org) and "Review" as the `resourceType` value. See the table under *Dataset Types and DataCite* for details and comparison. Please note that we are not using the "PeerReview" for `resourceTypeGeneral` because it is (in our view) specific to scholarly communications and may carry related connotations.

The following table summaries how regular datasets compare to review datasets.

Table 1: Differences between regular and review datasets

	Regular Dataset	Review Dataset
Collections/Datasets/Files search facet	dataset	dataset
Dataset Type search facet	dataset	review
DataCite	See table under *Add Dataset Type*	See table under *Add Dataset Type*

Creating a Review Dataset

You can only create a review dataset if setup has already been done by a superuser. See *Configure a Collection for Review Datasets* for details.

Review Datasets can only be created via API. You have the following options:

- *Create a Dataset with a Dataset Type (Software, Workflow, Review, etc.)* (Native API)
  - Here is an example JSON file for reference: `dataset-create-review.json`.
- *Create a Dataset with a Dataset Type* (Semantic API)
- *Import a Dataset with a Dataset Type (Software, Workflow, Review, etc.)*

When creating a review dataset you will likely need to fill in required fields like `itemReviewedUrl` as well as fields from one or more "rubric" metadata blocks, as described above under *Review Dataset Overview*.

If you point `itemReviewedUrl` at the URL form of a dataset (e.g. https://doi.org/10.5072/FK2/ABCDEF) that is in the same Dataverse installation as the review dataset, the review dataset is considered a local review and can be listed using the *List Reviews* API endpoint. These reviews appear in the Croissant metadata export if you enable the *dataverse.feature.croissant-with-local-reviews* feature flag.

Dataset Types and DataCite

Adding certain dataset types will result in a value other than "Dataset" being sent to DataCite (if you use DataCite) as shown in the table below.

Table 2: Values sent to DataCite for resourceTypeGeneral by Dataset Type

Dataset Type	Value sent to DataCite
dataset	<resourceType resourceTypeGeneral="Dataset"/>
software	<resourceType resourceTypeGeneral="Software"/>
workflow	<resourceType resourceTypeGeneral="Workflow"/>
review	<resourceType resourceTypeGeneral="Other">Review</resourceType>

Note that the value for resourceType (which is either empty or "Review", as shown above) can be overridden by values in the "Data Type" (`kindOfData`) metadata field.

2.5 Tabular Data File Ingest

Contents:

2.5.1 Supported File Formats

Tabular Data ingest supports the following file formats:

File format	Versions supported
SPSS (POR and SAV formats)	7 to 22
STATA	4 to 17 (see the Stata subsection)
R	up to 3
Excel	XLSX only (XLS is NOT supported)
CSV (comma-separated values)	(limited support)

See the subsections in the left sidebar for more information on each of these supported formats.

2.5.2 Tabular Data, Representation, Storage and Ingest

This section explains the basics of how tabular data is handled in the application and what happens during the ingest process, as the files uploaded by the user are processed and converted into the archival format in the Dataverse application.

Contents:

- *What Happens During this "Ingest"?*
- *Tabular Data and Metadata*
  - *Data vs. Metadata*
  - *Tabular Metadata in the Dataverse Software*
- *Uningest and Reingest*

What Happens During this "Ingest"?

The goal of our ingest process is to extract the data content from the user's files and archive it in an application-neutral, easily-readable format. What does this mean? - Commercial applications such as SPSS and Stata use their own, proprietary formats to encode their files. Some companies publish the specifications of their formats (Thank you Stata - much appreciated!), some don't (SPSS - yes, we are still frowning at you here at the Dataverse Project). Either way, reading these specially-formatted files requires some extra knowledge or special software. For these reasons they are not considered ideal for the purposes of archival preservation. The Dataverse installation stores the raw data content extracted from such files in plain text, TAB-delimited files. The metadata information that describes this content is stored separately, in a relational database, so that it can be accessed efficiently by the application. For the purposes of archival preservation it can be exported, in plain text XML files, using a standardized, open DDI Codebook format. (more info below)

Tabular Data and Metadata

Data vs. Metadata

A simple example is a numeric data column in a user's Stata file that contains 0s and 1s. These numeric values will be extracted and stored in a TAB-delimited file. By themselves, if you don't know what these values represent, these 1s and 0s are not meaningful data. So the Stata file has some additional information that describes this data vector: it represents the observation values of a *variable* with the *name* "party"; with a descriptive *label* "Party Affiliation"; and the 2 numeric values have *categorical labels* of "Democrat" for 0 and "Republican" for 1. This extra information that adds value to the data is *metadata*.

Tabular Metadata in the Dataverse Software

The structure of the metadata defining tabular data variables used in the Dataverse Software was originally based on the DDI Codebook format.

You can see an example of DDI output under the *Data Variable Metadata Access* section of the *Data Access API* section of the API Guide.

Uningest and Reingest

Ingest will only work for files whose content can be interpreted as a table. Multi-sheet spreadsheets and CSV files with a different number of entries per row are two examples where ingest will fail. This is non-fatal. The Dataverse software will not produce a .tab version of the file and will show a warning to users who can see the draft version of the dataset containing the file that will indicate why ingest failed. When the file is published as part of the dataset, there will be no indication that ingest was attempted and failed.

If the warning message is a concern, the Dataverse software includes both an API call (see *Uningest a File* in the *Native API* guide) and an Edit/Uningest menu option displayed on the file page, that allow a file to be uningested by anyone who can publish the dataset.

Uningest will remove the warning. Uningest can also be done for a file that was successfully ingested. This is only available to superusers. This will remove the variable-level metadata and the .tab version of the file that was generated.

If a file is a tabular format but was never ingested, .e.g. due to the ingest file size limit being lower in the past, or if ingest had failed, e.g. in a prior Dataverse version, an reingest API (see *Reingest a File* in the *Native API* guide) and a file page Edit/Reingest option in the user interface allow ingest to be tried again. As with Uningest, this fucntionality is only available to superusers.

2.5.3 SPSS

SPSS data files (POR and SAV formats).

Contents:

- *Supported Versions*
- *Limitations*
- *SPSS Control Cards - not supported*
- *Support for Language Encodings in SPSS*

Supported Versions

The Dataverse Software supports reading of all SPSS versions 7 to 22. But please see the "Limitations" section.

Limitations

SPSS does not openly publish the specifications of their proprietary file formats. Our ability to read and parse their files is based on some documentation online from unofficial sources, and some reverse engineering. Because of that we cannot, unfortunately, guarantee to be able to process *any* SPSS file uploaded.

However, we've been improving this process for a few years by now, and it should be quite robust in the current version of the Dataverse Software. Thus your chances of success - uploading an SPSS files and having it turned into a fully functional tabular data table in the Dataverse installation - should be reasonably good.

If you are having a problem with a particular SPSS file, please contact our support and we'll see if it's something we could further improve in our SPSS ingest driver.

SPSS Control Cards - not supported

In the past, there have been attempts to support SPSS "control cards", in addition to the .SAV and .POR files; both in the early VDC software, and in some versions of DVN. A "control card" is a plain text file with a set of SPSS commands that describe the raw data, provided in a separate, fixed-width-columns or comma-separated-values file. For various reasons, it has never been very successful.

Please contact us if you have any questions and/or strong feelings on this issue, or if you have serious amounts of data exclusively available in this format that you would like to ingest under the Dataverse Software.

Support for Language Encodings in SPSS

Historically, there was no support for specifying a particular language/code page encoding for the data stored in an SPSS file. Meaning, text values in none-ASCII encodings, or non-Latin characters could be entered and stored, but there was no setting to unambiguously specify what language, or what character set it was. By default, the Dataverse Software will try to interpret binary characters as UTF8. If that's not working - for example, if the descriptive labels and/or categorical values ingest as garbage - and if you happen to know what encoding was used in the original file, you can now specify it in the Ingest Options.

For example, if you know that the text in your SAV file is in Mandarin, and is encoded using the GB2312, specify it as follows:

Upload your file, in the "Edit Files" tab of the Dataset page. Once the file is recognized as SPSS/save, and *before* you click Save, go into the "Advanced Ingest Options", and select "Simplified Chinese, GB2312" in the nested menu under "Language Encoding" -> "East Asian".

2.5.4 Stata

Contents:

- *Supported Format Versions*

Of all the third party statistical software providers, Stata does the best job at documenting the internal format of their files, by far. And at making that documentation freely and easily available to developers (yes, we are looking at you, SPSS). Because of that, Stata is the best supported format for tabular data ingest.

Supported Format Versions

Of the **"New Stata dta"** formats (variations of the format in use since Stata 13) our ingest supports the following:

Stata format name	Introduced in	Used by
dta_117	Stata 13	Stata 13
dta_118	Stata 14	Stata 14 - 19
dta_119	Stata 15	Stata 15 - 19

This means that, in theory, every dta file produced by Stata v.13 - 17 should be ingestible. (Please see below for more information on Stata 18 and 19). In practice, we cannot *guarantee* that our code will in fact be able to parse any such file. There is always a possibility that we missed a certain way to compose the data that the ingest will fail to understand. So, if you encounter such an error, where Dataverse **tries but fails** to ingest a Stata file in one of these 3 formats, please open a GitHub issue and we will try to address it. Please note that this a different scenario from when Dataverse skips even trying to ingest a file (with no ingest errors shown in the UI). As that will in most cases be the result of the file exceeding the size limit set by the Dataverse instance administrators, or a client uploading the file with a wrong content type attached, so that Dataverse fails to recognize it as Stata.

Please note that there was an issue present in older versions of Dataverse where Stata 13-17 files were not ingested when deposited via direct upload to S3. The issue was accompanied by the confusing error message `The file is not in a STATA format that we can read or support` shown in the UI. Fortunately, a case like this can be addressed by running the reIngest API on the affected file.

The following 2 formats have been introduced in 2024 and are **not yet supported**:

Stata format name	Introduced in	Used by
dta_120	Stata 18	Stata 18 - 19
dta_121	Stata 18	Stata 18 - 19

Please note however, that this does not mean that no files produced by Stata 18 or 19 are ingestable! In reality, in most cases these versions of Stata still save files in the `dta_118` (i.e., Stata 14) format, with the later formats only utilized when necessary. When, for example, the number of variables in the datafile exceeds what `dta_118` can handle, or when it has "alias variables" introduced in Stata 18. Case in point, in a year since the introduction of these 2 newest formats, it appears that not a single file in either of them has been uploaded on production Dataverse instance at IQSS. We are planning to eventually add support for these formats, but it is not considered a priority as of yet. However, please feel free to open a GitHub issue if this is an important use case for you.

"Old Stata", a distinctly different format used by Stata versions prior to 13 is supported. However, this functionality is considered legacy code that we no longer actively maintain. If any problems or bugs are found in it, we cannot promise that the core development team will be able to prioritize looking into such. We will of course gladly accept a properly submitted pull request from the user community.

2.5.5 R Data Format

Support for R (.RData) files has been introduced in DVN 3.5.

Contents:

- *Overview*
- *Data Formatting Requirements*
- *Data Types, compared to other supported formats (Stat, SPSS)*
  - *Integers, Doubles, Character strings*
  - *R Factors*
  - *Boolean values*
  - *Limitations of R, as compared to SPSS and STATA*
- *Time values in R*

Overview

R has been increasingly popular in the research/academic community, owing to the fact that it is free and open-source (unlike SPSS and STATA). Consequently, there is an increasing amount of data available exclusively in R format.

Data Formatting Requirements

The data must be formatted as an R dataframe (`data.frame()`). If an .RData file contains multiple dataframes, only the first one will be ingested (this may change in the future).

Data Types, compared to other supported formats (Stat, SPSS)

Integers, Doubles, Character strings

The handling of these types is intuitive and straightforward. The resulting tab file columns, summary statistics and UNF signatures should be identical to those produced by ingesting the same vectors from SPSS and Stata.

Things that are unique to R:

R explicitly supports Missing Values for all of the types above; Missing Values encoded in R vectors will be recognized and preserved in TAB files, counted in the generated summary statistics and data analysis. Please note however that the Dataverse Software notation for a missing value, as stored in a TAB file, is an empty string, an not "NA" as in R.

In addition to Missing Values, R recognizes "Not a Value" (NaN) and positive and negative infinity for floating point variables. These are now properly supported by the Dataverse Software.

Also note, that unlike Stata, that does recognize "float" and "double" as distinct data types, all floating point values in R are in fact doubles.

R Factors

These are ingested as "Categorical Values" in the Dataverse installation.

One thing to keep in mind: in both Stata and SPSS, the actual value of a categorical variable can be both character and numeric. In R, all factor values are strings, even if they are string representations of numbers. So the values of the resulting categoricals in the Dataverse installation will always be of string type too.

Another thing to note is that R factors have no builtin support for SPSS or STATA-like descriptive labels. This is in fact potentially confusing, as they also use the word "label", in R parlance. However, in the context of a factor in R, it still refers to the "payload", or the data content of its value. For example, if you create a factor with the "labels" of *democrat*, *republican* and *undecided*, these strings become the actual values of the resulting vector. Once ingested in the Dataverse installation, these values will be stored in the tab-delimited file. The Dataverse Software DataVariable object representing the vector will be of type "Character" and have 3 VariableCategory objects with the *democrat*, etc. for **both** the CategoryValue and CategoryLabel. (In one of the future releases, we are planning to make it possible for the user to edit the CategoryLabel, using it for its intended purpose - as a descriptive, human-readable text text note).

To properly handle R vectors that are *ordered factors* the Dataverse Software (starting with DVN 3.6) supports the concept of an "Ordered Categorical" - a categorical value where an explicit order is assigned to the list of value labels.

Boolean values

R Boolean (logical) values are supported.

Limitations of R, as compared to SPSS and STATA

Most noticeably, R lacks a standard mechanism for defining descriptive labels for the data frame variables. In the Dataverse Software, similarly to both Stata and SPSS, variables have distinct names and labels; with the latter reserved for longer, descriptive text. With variables ingested from R data frames the variable name will be used for both the "name" and the "label".

Optional R packages exist for providing descriptive variable labels; in one of the future versions support may be added for such a mechanism. It would of course work only for R files that were created with such optional packages.

Similarly, R categorical values (factors) lack descriptive labels too. **Note:** This is potentially confusing, since R factors do actually have "labels". This is a matter of terminology - an R factor's label is in fact the same thing as the "value" of a categorical variable in SPSS or Stata and the Dataverse Software; it contains the actual meaningful data for the given observation. It is NOT a field reserved for explanatory, human-readable text, such as the case with the SPSS/Stata "label".

Ingesting an R factor with the level labels "MALE" and "FEMALE" will produce a categorical variable with "MALE" and "FEMALE" in the values and labels both.

Time values in R

This warrants a dedicated section of its own, because of some unique ways in which time values are handled in R.

R makes an effort to treat a time value as a real time instance. This is in contrast with either SPSS or Stata, where time value representations such as "Sep-23-2013 14:57:21" are allowed; note that in the absence of an explicitly defined time zone, this value cannot be mapped to an exact point in real time. R handles times in the "Unix-style" way: the value is converted to the "seconds-since-the-Epoch" Greenwich time (GMT or UTC) and the resulting numeric value is stored in the data file; time zone adjustments are made in real time as needed.

Things still get ambiguous and confusing when R **displays** this time value: unless the time zone was explicitly defined, R will adjust the value to the current time zone. The resulting behavior is often counter-intuitive: if you create a time value, for example:

```
timevalue<-as.POSIXct("03/19/2013 12:57:00", format = "%m/%d/%Y %H:%M:%OS");
```

on a computer configured for the San Francisco time zone, the value will be differently displayed on computers in different time zones; for example, as "12:57 PST" while still on the West Coast, but as "15:57 EST" in Boston.

If it is important that the values are always displayed the same way, regardless of the current time zones, it is recommended that the time zone is explicitly defined. For example:

```
attr(timevalue,"tzone")<-"PST"
```

or

```
timevalue<-as.POSIXct("03/19/2013 12:57:00", format = "%m/%d/%Y %H:%M:%OS", tz="PST");
```

Now the value will always be displayed as "15:57 PST", regardless of the time zone that is current for the OS ... **BUT ONLY** if the OS where R is installed actually understands the time zone "PST", which is not by any means guaranteed! Otherwise, it will **quietly adjust** the stored GMT value to **the current time zone**, yet it will still display it with the "PST" tag attached!** One way to rephrase this is that R does a fairly decent job **storing** time values in a non-ambiguous, platform-independent manner - but gives you no guarantee that the values will be displayed in any way that is predictable or intuitive.

In practical terms, it is recommended to use the long/descriptive forms of time zones, as they are more likely to be properly recognized on most computers. For example, "Japan" instead of "JST". Another possible solution is to explicitly use GMT or UTC (since it is very likely to be properly recognized on any system), or the "UTC+<OFFSET>" notation. Still, none of the above **guarantees** proper, non-ambiguous handling of time values in R data sets. The fact that R **quietly** modifies time values when it doesn't recognize the supplied timezone attribute, yet still appends it to the **changed** time value does make it quite difficult. (These issues are discussed in depth on R-related forums, and no attempt is made to summarize it all in any depth here; this is just to made you aware of this being a potentially complex issue!)

An important thing to keep in mind, in connection with the Dataverse Software ingest of R files, is that it will **reject** an R data file with any time values that have time zones that we can't recognize. This is done in order to avoid (some) of the potential issues outlined above.

It is also recommended that any vectors containing time values ingested into the Dataverse installation are reviewed, and the resulting entries in the TAB files are compared against the original values in the R data frame, to make sure they have been ingested as expected.

Another **potential issue** here is the **UNF**. The way the UNF algorithm works, the same date/time values with and without the timezone (e.g. "12:45" vs. "12:45 EST") **produce different UNFs**. Considering that time values in Stata/SPSS do not have time zones, but ALL time values in R do (yes, they all do - if the timezone wasn't defined explicitly, it implicitly becomes a time value in the "UTC" zone!), this means that it is **impossible** to have 2 time value vectors, in Stata/SPSS and R, that produce the same UNF.

A pro tip: if it is important to produce SPSS/Stata and R versions of the same data set that result in the same UNF when ingested, you may define the time variables as **strings** in the R data frame, and use the "YYYY-MM-DD HH:mm:ss"

formatting notation. This is the formatting used by the UNF algorithm to normalize time values, so doing the above will result in the same UNF as the vector of the same time values in Stata.

Note: date values (dates only, without time) should be handled the exact same way as those in SPSS and Stata, and should produce the same UNFs.

2.5.6 Excel

Microsoft Excel files (XLSX format).

Contents:

- *Supported Formats*
- *Limitations*
- *Ingest Errors*
- *Example Data*

Supported Formats

Only the newer XLSX Excel files are supported. We are not planning to add support for the old-style, binary XLS files.

Limitations

If an Excel file has multiple sheets, only the first sheet of the file will be ingested. The other sheets will be available when a user downloads the original Excel file. To have all sheets of an Excel file ingested and searchable at the variable level, upload each sheet as an individual file in your dataset.

Ingest Errors

You may encounter ingest errors after uploading an Excel file if the file is formatted in a way that can't be ingested by the Dataverse software. Ingest errors can be caused by a variety of formatting inconsistencies, including:

- line breaks in a cell
- blank cells
- single cells that span multiple rows
- missing headers

Example Data

An example of an Excel file that successfully ingests is available in the Dataverse Sample Data GitHub repository.

2.5.7 CSV/TSV

Contents:

- *Ingest of Comma-Separated Values and Tab-Separated Values files as tabular data*
- *Main formatting requirements*
- *Limitations*
- *Recognized data types and formatting*

Ingest of Comma-Separated Values and Tab-Separated Values files as tabular data

The Dataverse installation will make an attempt to turn CSV and TSV files uploaded by the user into tabular data, using the Apache CSV parser.

Main formatting requirements

The first row in the document will be treated as the CSV's header, containing variable names for each column.

Each following row must contain the same number of comma-separated values ("cells") as that header.

As of the Dataverse Software 4.8 release, we allow ingest of CSV files with commas and line breaks within cells. A string with any number of commas and line breaks enclosed within double quotes is recognized as a single cell. Double quotes can be encoded as two double quotes in a row ("").

For example, the following lines:

```
a,b,"c,d
efgh""ijk""l",m,n
```

are recognized as a **single** row with **5** comma-separated values (cells):

```
a
b
c,d\nefgh"ijk"l
m
n
```

(where \n is a new line character)

Limitations

Compared to other formats, relatively little information about the data ("variable-level metadata") can be extracted from a CSV file. Aside from the variable names supplied in the top line, the ingest will make an educated guess about the data type of each comma-separated column. One of the supported rich file formats (Stata, SPSS and R) should be used if you need to provide more descriptive variable-level metadata (variable labels, categorical values and labels, explicitly defined data types, etc.).

Recognized data types and formatting

The application will attempt to recognize numeric, string, and date/time values in the individual columns.

For dates, the `yyyy-MM-dd` format is recognized.

For date-time values, the following 2 formats are recognized:

`yyyy-MM-dd HH:mm:ss`

`yyyy-MM-dd HH:mm:ss z` (same format as the above, with the time zone specified)

For numeric variables, the following special values are recognized:

`inf`, `+inf` - as a special IEEE 754 "positive infinity" value;

`NaN` - as a special IEEE 754 "not a number" value;

An empty value (i.e., a comma followed immediately by another comma, or the line end), or `NA` - as a *missing value*.

`null` - as a numeric *zero*.

(any combinations of lower and upper cases are allowed in the notations above).

In character strings, an empty value (a comma followed by another comma, or the line end) is treated as an empty string (NOT as a *missing value*).

Any non-Latin characters are allowed in character string values, **as long as the encoding is UTF8**.

Note: When the ingest recognizes a CSV or TSV column as a numeric vector, or as a date/time value, this information is reflected and saved in the database as the *data variable metadata*. To inspect that metadata, select *Variable Metadata* listed as a download option for the tabular file. This will export the variable records in the DDI XML format. (Alternatively, this metadata fragment can be downloaded via the Data Access API; for example: `http://localhost:8080/api/access/datafile/<FILEID>/metadata/ddi`).

The most immediate implication is in the calculation of the UNF signatures for the data vectors, as different normalization rules are applied to numeric, character, and date/time values. (see the *Universal Numerical Fingerprint (UNF)* section for more information). If it is important to you that the UNF checksums of your data are accurately calculated, check that the numeric and date/time columns in your file were recognized as such (as `type=numeric` and `type=character, category=date(time)`, respectively). If, for example, a column that was supposed to be numeric is recognized as a vector of character values (strings), double-check that the formatting of the values is consistent. Remember, a single improperly-formatted value in the column will turn it into a vector of character strings, and result in a different UNF. Fix any formatting errors you find, delete the file from the dataset, and try to ingest it again.

2.6 Appendix

Additional documentation complementary to the User Guide.

Contents:

- *Metadata References*
  - *Supported Metadata*
  - *Experimental Metadata*
  - *See Also*

2.6.1 Metadata References

The Dataverse Project is committed to using standard-compliant metadata to ensure that a Dataverse installation's metadata can be mapped easily to standard metadata schemas and be exported into JSON format (XML for tabular file metadata) for preservation and interoperability.

Supported Metadata

Detailed below are what metadata schemas we support for Citation and Domain Specific Metadata in the Dataverse Project:

- Citation Metadata (see .tsv): compliant with DDI Lite, DDI 2.5 Codebook, DataCite 4.5, and Dublin Core's DCMI Metadata Terms . Language field uses ISO 639-1 controlled vocabulary.
- Geospatial Metadata (see .tsv): compliant with ISO 19115, DDI Lite, DDI 2.5 Codebook, DataCite 4.5, and Dublin Core. Country / Nation field uses ISO 3166-1 controlled vocabulary.
- Social Science & Humanities Metadata (see .tsv): compliant with DDI Lite, DDI 2.5 Codebook, and Dublin Core.
- Astronomy and Astrophysics Metadata (see .tsv): These metadata elements can be mapped/exported to the International Virtual Observatory Alliance's (IVOA) VOResource Schema format and is based on Virtual Observatory (VO) Discovery and Provenance Metadata.
- Life Sciences Metadata (see .tsv): based on ISA-Tab Specification, along with controlled vocabulary from subsets of the OBI Ontology and the NCBI Taxonomy for Organisms.
- Journal Metadata (see .tsv): based on the Journal Archiving and Interchange Tag Set, version 1.2.
- 3D Objects Metadata (see .tsv).

Experimental Metadata

Unlike supported metadata, experimental metadata is not enabled by default in a new Dataverse installation. Feedback via any channel is welcome!

- CodeMeta Software Metadata: based on the CodeMeta Software Metadata Schema, version 2.0 (see .tsv version)
- Computational Workflow Metadata (see .tsv): adapted from Bioschemas Computational Workflow Profile, version 1.0 and Codemeta.
- Archival Metadata (see .tsv): Enables repositories to register metadata relating to the potential archiving of the dataset at a depositor archive, whether that be your own institutional archive or an external archive, i.e. a historical archive.
- Local Contexts Metadata (see .tsv): Supports integration with the Local Contexts platform, enabling the use of Traditional Knowledge and Biocultural Labels, and Notices. For more information on setup and configuration, see *LocalContexts Integration*.
- Review Metadata (see .tsv): For *Review Datasets*.

Please note: these custom metadata schemas are not included in the Solr schema for indexing by default, you will need to add them as necessary for your custom metadata blocks. See "Update the Solr Schema" in *Metadata Customization*.

See Also

See also the Dataverse Software 4.0 Metadata Crosswalk: DDI, DataCite, DC, DCTerms, VO, ISA-Tab document and the *Metadata Customization* section of the Admin Guide.

CHAPTER

THREE

ADMIN GUIDE

This guide documents the functionality only available to superusers (such as "dataverseAdmin") through the *Dashboard* as well as command line utilities of interest to sysadmins.

Contents:

3.1 Features

This page highlights features for administrators and power-users of a Dataverse installation.

See What is Dataverse? *to learn about its* Core Capabilities *for researchers if you're new to Dataverse.*

- *Artifical Intelligence*
- *Access and Download*
- *Administration*
- *Authentication*
- *Customization*
- *FAIR Data Publication*
- *File Management*
- *Geospatial Data Support*
- *Integrations*
- *Interoperability*
- *Reusability*

Artifical Intelligence

AI Tools A number of AI tools integrate with Dataverse.

More information.

Model Context Protocol Model Context Protocol (MCP) is a standard for AI Agents to communicate with tools and services.

More information.

Access and Download

Faceted Search Facets are data driven and customizable per collection.

More information.

File Previews A preview is available for text, tabular, image, audio, video, and geospatial files.

More information.

Preview URL Create a URL for reviewers to view an unpublished (and optionally anonymized) dataset.

More information.

Guestbook Optionally collect data about who is downloading the files from your datasets.

More information.

Download in Open Tabular Formats Proprietary tabular formats are converted into TSV and RData for download.

More information.

Administration

User Management Dashboard for common user-related tasks.

More information.

Quotas For number of files, amount of storage, etc.

More information.

Usage Statistics and Metrics Download counters, support for Make Data Count.

More information.

Configurable Notifications In-app and email notifications for access requests, requests for review, etc. can be muted.

More information.

Authentication

Login via Shibboleth Single Sign On (SSO) using your institution's credentials.

More information.

Login via ORCID, Google, GitHub, or Microsoft Log in using popular OAuth2 providers.

More information.

Login via OpenID Connect (OIDC) Log in using your institution's identity provider or a third party.

More information.

Customization

Branding Your installation can be branded with a custom homepage, header, footer, CSS, etc.

More information.

Internationalization The Dataverse software has been translated into multiple languages.

More information.

Customization of Collections Each personal or organizational collection can be customized and branded.

More information.

Widgets Embed listings of data in external websites.

More information.

FAIR Data Publication

Support for FAIR Data Principles Findable, Accessible, Interoperable, Reusable.

More information.

Versioning History of changes to datasets and files are preserved.

More information.

Prepublication Review Support Datasets start as drafts and can be submitted for review before publication where curators can mark datasets with curation status labels.

More information.

Labels for Traditional Knowledge Integrate with the Local Contexts platform, enabling the use of Traditional Knowledge and Biocultural Labels, and Notices.

More information.

Locally FAIR Data (Experimental) Some datasets' existence cannot be shared with the public right away, but should be findable and accessible to certain groups of people.

More information.

File Management

File Hierarchy Users are able to control dataset file hierarchy and directory structure.

More information.

Restricted Files Control who can download files and choose whether or not to enable a "Request Access" button.

More information.

Embargo Make files inaccessible until an embargo end date.

More information.

Retention Periods Make files inaccessible once the retention period set has passed.

More information.

Metadata Extraction from Files Populate dataset metadata fields from tabular, NetCDF, HDF5, and FITS files.

More information.

Configurable Storage Choose between filesystem or object storage, configurable per collection and per dataset.

More information.

Direct Upload and Download for S3 After a permission check, files can pass freely and directly between a client computer and S3.

More information.

Fixity Checks for Files MD5, SHA-1, SHA-256, SHA-512, UNF.

More information.

Auxiliary Files for Data Files Each data file can have any number of auxiliary files for documentation or other purposes (experimental).

More information.

Geospatial Data Support

Geospatial Metadata Fields There is a dedicated geospatial metadata block.

More information.

Geospatial File Preview GeoJSON, GeoTIFF, and Shapefiles can be previewed as a map.

More information.

Geospatial Search API Pass `geo_point` and `geo_radius` to find datasets based on their bounding box.

More information.

Integrations

DataCite DOIs are reserved, and when datasets are published, their metadata is published to DataCite.

More information.

Handle Handles are a Persistent ID (PID) that are an alternative to DOIs.

More information.

Globus Upload from and download to Dataverse using Globus endpoints.

More information.

RSpace Exchange data and metadata with RSpace (e.g. IGSN ID). For example, a Data Management Plan (DMP) can be uploaded to RSpace and updated with the DOI of a Dataverse dataset.

More information.

GitHub A GitHub Action is available to upload files from GitHub to a dataset.

More information.

iRODS Pull data from an iRODS instance to a Dataverse dataset.

More information.

Dropbox Upload files stored on Dropbox.

More information.

Jupyter Notebooks Datasets can be opened in Binder to run code in Jupyter notebooks, RStudio, and other computation environments. They can also be previewed in Dataverse itself.

More information.

Galaxy Import files directly from Dataverse into Galaxy as well as publish datasets containing artifacts (Histories, datasets, etc.) from Galaxy to Dataverse.

More information.

External Tools Enable additional features not built in to the Dataverse software.

More information.

Additional Integrations Dataverse integrates with a wide variety of third party systems, some of which are highlighted above.

For a full list, see *Integrations*.

Interoperability

APIs Search API, Data Deposit API, Data Access API, Metrics API, Migration API, etc. and client libraries in various languages.

More information.

OAI-PMH Metadata Harvesting Serve and harvest metadata to and from other systems (e.g. DataCite, other Dataverse installations, etc.) using standardized metadata formats.

More information.

Schema.org JSON-LD Used by Google Dataset Search and other services for discoverability.

More information.

Croissant Export metadata as linked data following the Croissant ontology.

More information.

Signposting Enable easier machine access to datasets by adding linkset in a Dataverse header.

More information.

External Vocabulary Let users pick from external vocabularies (provided via API/SKOSMOS) when filling in metadata.

More information.

BagIt Export For preservation, bags can be sent to the local filesystem, Duracloud, and Google Cloud.

More information.

RO-Crate Export dataset metadata as an ro-crate.json.

More information.

Reusability

Multiple License Support Users can select from multiple standard and provided custom licenses.

More information.

Custom Terms of Use Users can write custom terms of use in place of a predefined license.

More information.

Data Citation Formats EndNote XML, RIS, BibTeX, or 1000+ CSL formats at the dataset or file level.

More information.

Provenance At the file level, upload standard W3C provenance files or enter free text instead.

More information.

Post-Publication Workflows Allow publication of a dataset to trigger external processes and integrations.

More information.

3.2 Dashboard

The Dataverse Software offers a dashboard of administrative tools for superusers only. If you are a logged-in superuser, you can access it by clicking your username in the navbar, and then clicking "Dashboard" from the dropdown. You can verify that you are a superuser by checking the color of your username in the navbar. If it's red, you have the right permissions to use the Dashboard. Superusers can give other users the superuser status via *User Administration*.

Contents:

- *Harvesting*
  - *Harvesting Clients*
  - *Harvesting Servers*
- *Metadata Export*
- *Users*
- *Move Data*

3.2.1 Harvesting

Harvesting Clients

This dashboard tool allows you to set up which other repositories your Dataverse installation harvests metadata records from. You can see a list of harvesting clients and add, edit, or remove them. See the *Managing Harvesting Clients* section for more details.

Harvesting Servers

This dashboard tool allows you to define sets of local datasets to make available to remote harvesting clients. You can see a list of sets and add, edit, or remove them. See the *Managing Harvesting Server and Sets* section for more details.

3.2.2 Metadata Export

This part of the Dashboard is simply a reminder message that metadata export happens through the Dataverse Software API. See the *Metadata Export* section and the *Native API* section of the API Guide for more details.

3.2.3 Users

This dashboard tool allows you to search a list of all users of your Dataverse installation. You can remove roles from user accounts and assign or remove superuser status. See the *User Administration* section for more details.

3.2.4 Move Data

This tool allows you to move datasets. To move Dataverse collections, see the *Managing Datasets and Dataverse Collections* section.

3.3 External Tools

External tools can provide additional features that are not part of the Dataverse Software itself, such as data file previews, visualization, and curation.

Contents:

- *Inventory of External Tools*
- *Managing External Tools*
  - *Adding External Tools to a Dataverse Installation*
  - *Listing All External Tools in a Dataverse Installation*
  - *Showing an External Tool in a Dataverse Installation*
  - *Removing an External Tool From a Dataverse Installation*
- *Testing External Tools*
  - *File Level vs. Dataset Level*
  - *File Level Explore Tools*
  - *File Level Preview Tools*
  - *File Level Query Tools*
  - *File Level Configure Tools*
  - *Dataset Level Explore Tools*
  - *Dataset Level Configure Tools*

- *Writing Your Own External Tool*

3.3.1 Inventory of External Tools

Tool	Type	Sco	Description
Data Explorer	explore	file	A GUI which lists the variables in a tabular data file allowing searching, charting and cross tabulation analysis. The latest version incorporates the Data Curation Tool, a GUI for curating data by adding labels, groups, weights and other details to assist with informed reuse. See the README.md file at https://github.com/scholarsportal/Dataverse-Data-Explorer for the instructions on adding Data Explorer to your Dataverse.
Binder	explore	data	Binder allows you to spin up custom computing environments in the cloud (including Jupyter notebooks) with the files from your dataset. See https://github.com/IQSS/dataverse-binder-redirect for installation instructions.
File Previewers	explore	file	A set of tools that display the content of files - including audio, html, Hypothes.is annotations, images, PDF, Markdown, text, video, tabular data, spreadsheets, GeoJSON, zip, and NcML files - allowing them to be viewed without downloading the file. The previewers can be run directly from github.io, so the only required step is using the Dataverse API to register the ones you want to use. Documentation, including how to optionally brand the previewers, and an invitation to contribute through github are in the README.md file. Initial development was led by the Qualitative Data Repository and the spreasdheet previewer was added by the Social Sciences and Humanities Open Cloud (SSHOC) project. https://github.com/gdcc/dataverse-previewers
Ask the Data	query	file	Ask the Data is an experimental tool that allows you ask natural language questions about the data contained in Dataverse tables (tabular data). See the README.md file at https://github.com/IQSS/askdataverse/tree/main/askthedata for the instructions on adding Ask the Data to your Dataverse installation.
TurboCurator by ICPSR	configure	data	TurboCurator generates metadata improvements for title, description, and keywords. It relies on open AI's ChatGPT & ICPSR best practices. See the TurboCurator Dataverse Administrator page for more details on how it works and adding TurboCurator to your Dataverse installation.
JupyterHub	explore	file	The Dataverse-to-JupyterHub Data Transfer Connector is a tool that simplifies the transfer of data between Dataverse repositories and the cloud-based platform JupyterHub. It is designed for researchers, scientists, and data analysts, facilitating collaboration on projects by seamlessly moving datasets and files. The tool is a lightweight client-side web application built using React and relies on the Dataverse External Tool feature, allowing for easy deployment on modern integration systems. Currently optimized for small to medium-sized files, future plans include extending support for larger files and signed Dataverse endpoints. For more details, you can refer to the external tool manifest: https://forge.inrae.fr/dipso/eosc-pillar/dataverse-jupyterhub-connector/-/blob/master/externalTools.json
3DViewer by openforestdata.pl	explore	file	The 3DViewer by openforestdata.pl can be used to explore 3D files (e.g. STL format). It was presented by Kamil Guryn during the 2020 community meeting (slide deck, video) and can be found at https://github.com/OpenForestData/open-forest-data-previewers
Open OnDemand	explore	data	Open OnDemand is a web frontend to High Performance Computing (HPC) resources that is integrated with Dataverse. See https://iqss.github.io/ondemand-loop/ for installation instructions.

3.3.2 Managing External Tools

Adding External Tools to a Dataverse Installation

To add an external tool to your Dataverse installation you must first download a JSON file for that tool, which we refer to as a "manifest". It should look something like this:

```
{
  "displayName": "Fabulous File Tool",
  "description": "A non-existent tool that is fabulous fun for files!",
  "toolName": "fabulous",
  "scope": "file",
  "types": [
    "explore",
    "preview"
  ],
  "toolUrl": "https://fabulousfiletool.com",
  "contentType": "text/tab-separated-values",
  "httpMethod":"GET",
  "toolParameters": {
    "queryParameters": [
      {
        "fileid": "{fileId}"
      },
      {
        "datasetPid": "{datasetPid}"
      },
      {
        "locale":"{localeCode}"
      }
    ]
  },
  "allowedApiCalls": [
    {
      "name":"retrieveDataFile",
      "httpMethod":"GET",
      "urlTemplate":"/api/v1/access/datafile/{fileId}",
      "timeOut":270
    }
  ]
}
```

Go to *Inventory of External Tools* and download a JSON manifest for one of the tools by following links in the description to installation instructions.

Configure the tool with the curl command below, making sure to replace the `fabulousFileTool.json` placeholder for name of the JSON manifest file you downloaded.

```
curl -X POST -H 'Content-type: application/json' http://localhost:8080/api/admin/
↪externalTools --upload-file fabulousFileTool.json
```

This API is Superuser only. Note the endpoint difference (/api/externalTools instead of /api/admin/externalTools).

```
curl -s -H "X-Dataverse-key:$API_TOKEN" -X POST -H 'Content-type: application/json'␣
↪http://localhost:8080/api/externalTools --upload-file fabulousFileTool.json
```

Listing All External Tools in a Dataverse Installation

To list all the external tools that are available in a Dataverse installation:

```
curl http://localhost:8080/api/admin/externalTools
```

This API is open to any user. Note the endpoint difference (/api/externalTools instead of /api/admin/externalTools).

```
curl http://localhost:8080/api/externalTools
```

Showing an External Tool in a Dataverse Installation

To show one of the external tools that are available in a Dataverse installation, pass its database id:

```
export TOOL_ID=1
curl http://localhost:8080/api/admin/externalTools/$TOOL_ID
```

This API is open to any user. Note the endpoint difference (/api/externalTools instead of /api/admin/externalTools).

```
curl http://localhost:8080/api/externalTools/$TOOL_ID
```

Removing an External Tool From a Dataverse Installation

Assuming the external tool database id is "1", remove it with the following command:

```
export TOOL_ID=1
curl -X DELETE http://localhost:8080/api/admin/externalTools/$TOOL_ID
```

This API is Superuser only. Note the endpoint difference (/api/externalTools instead of /api/admin/externalTools).

```
curl -s -H "X-Dataverse-key:$API_TOKEN" -X DELETE http://localhost:8080/api/
↪externalTools/$TOOL_ID
```

3.3.3 Testing External Tools

Once you have added an external tool to your Dataverse installation, you will probably want to test it to make sure it is functioning properly.

File Level vs. Dataset Level

File level tools are specific to the file type (content type or MIME type). For example, a tool may work with PDFs, which have a content type of "application/pdf".

In contrast, dataset level tools are always available no matter what file types are within the dataset.

File Level Explore Tools

File level explore tools provide a variety of features from data visualization to statistical analysis.

For each supported file type, file level explore tools appear in the file listing of the dataset page as well as under the "Access" button on each file page.

File Level Preview Tools

File level preview tools allow the user to see a preview of the file contents without having to download it.

When a file has a preview available, a preview icon will appear next to that file in the file listing on the dataset page. On the file page itself, the preview will appear in a Preview tab (renamed File Tools, if multiple tools are available) either immediately or once a guestbook has been filled in or terms, if any, have been agreed to.

File Level Query Tools

File level query tools allow the user to ask questions (e.g. natural language queries) of a data table's contents without having to download it.

When a file has a query tool available, a query icon will appear next to that file in the file listing on the dataset page. On the file page itself, the query tool will appear in a Query tab (renamed File Tools, if multiple tools are available) either immediately or once a guestbook has been filled in or terms, if any, have been agreed to.

File Level Configure Tools

File level configure tools are only available when you log in and have write access to the file. The file type determines if a configure tool is available. For example, a configure tool may only be available for tabular files.

Dataset Level Explore Tools

Dataset level explore tools allow the user to explore all the files in a dataset.

Dataset Level Configure Tools

Dataset level configure tools can be launched by users who have edit access to the dataset. These tools are found under the "Edit Dataset" menu.

3.3.4 Writing Your Own External Tool

If you plan to write a external tool, see the *Building External Tools* section of the API Guide.

If you have an idea for an external tool, please let the Dataverse Project community know by posting about it on the dataverse-community mailing list: https://groups.google.com/forum/#!forum/dataverse-community

3.4 Discoverability

Datasets are made discoverable by a variety of methods.

Contents:

- *DataCite Integration*
- *OAI-PMH (Harvesting)*
- *Machine-Readable Metadata on Dataset Landing Pages*
  - *Dublin Core HTML Meta Tags*
  - *Croissant Metadata in the <head> of Dataset Landing Pages*
  - *Signposting*
- *Additional Discoverability Through Integrations*

3.4.1 DataCite Integration

If you are using DataCite as your DOI provider, when datasets are published, metadata is pushed to DataCite, where it can be searched. For more information, see *Legacy Single PID Provider: :DoiProvider* in the Installation Guide.

3.4.2 OAI-PMH (Harvesting)

The Dataverse software supports a protocol called OAI-PMH that facilitates harvesting dataset metadata from one system into another. For details on harvesting, see the *Managing Harvesting Server and Sets* section.

3.4.3 Machine-Readable Metadata on Dataset Landing Pages

As recommended in A Data Citation Roadmap for Scholarly Data Repositories, the Dataverse software embeds metadata on dataset landing pages in a variety of machine-readable ways.

Dublin Core HTML Meta Tags

The HTML source of a dataset landing page includes "DC" (Dublin Core) <meta> tags such as the following:

```
<meta name="DC.identifier" content="..."
<meta name="DC.type" content="Dataset"
<meta name="DC.title" content="..."
```

Croissant Metadata in the <head> of Dataset Landing Pages

Croissant is a metadata format for machine learning datasets.

In Dataverse, the <head> of the HTML source of a dataset landing page includes a "slim" version of Croissant metadata like this:

```
<script type="application/ld+json">{"@context":..."cr":"http://mlcommons.org/croissant/".
↪..
```

This slim version does not have any information about files or variables but is otherwise is identical to the Croissant file you can download from a dataset landing page by clicking "Metadata" then "Export Metadata" (see *Supported Metadata Export Formats*). From the API you can download both `croissant` and `croissantSlim` formats (see *Export Metadata of a Dataset in Various Formats*).

We include Croissant in the <head> because it's recommended by Google for Google Dataset Search, where they offer a filter to narrow results to only datasets with support for Croissant.

Before Croissant was invented, Google recommended a different format that Dataverse refers to as "Schema.org JSON-LD" in the user interface (and `schema.org` in the API). If you prefer to put that older format in the <head>, which was the behavior in older versions of Dataverse, see *dataverse.legacy.schemaorg-in-html-head*.

See also the *dataverse.feature.croissant-with-local-reviews* feature flag.

Signposting

The Dataverse software supports Signposting. This allows machines to request more information about a dataset through the Link HTTP header. Links to all enabled metadata export formats are given. See *Supported Metadata Export Formats* for a list.

There are 2 Signposting profile levels, level 1 and level 2. In this implementation,

- Level 1 links are shown as recommended in the "Link" HTTP header, which can be fetched by sending an HTTP HEAD request, e.g. `curl -I https://demo.dataverse.org/dataset.xhtml?persistentId=doi:10.5072/FK2/KPY4ZC`. The number of author and file links in the level 1 header can be configured as described below.
- The level 2 linkset can be fetched by visiting the dedicated linkset page for that artifact. The link can be seen in level 1 links with key name `rel="linkset"`.

Note: Authors without author link will not be counted nor shown in any profile/linkset. The following configuration options are available:

- *dataverse.signposting.level1-author-limit*

  Sets the max number of authors to be shown in *level 1* profile. If the number of authors (with identifier URLs) exceeds this value, no author links will be shown in *level 1* profile. The default is 5.

- *dataverse.signposting.level1-item-limit*

  Sets the max number of items/files which will be shown in *level 1* profile. Datasets with too many files will not show any file links in *level 1* profile. They will be shown in *level 2* linkset only. The default is 5.

See also *Retrieve Signposting Information* in the API Guide.

3.4.4 Additional Discoverability Through Integrations

See *Discoverability* in the Integrations section for additional discovery methods you can enable.

3.5 Managing Harvesting Clients

Contents:

- *Your Dataverse Installation as a Metadata Harvester*
- *Managing Harvesting Clients*
  - *Harvesting from Datacite*
  - *How to Stop a Harvesting Run in Progress*
  - *What if a Run Fails?*
- *Harvesting Client Changelog*
- *Harvesting Non-OAI-PMH*

3.5.1 Your Dataverse Installation as a Metadata Harvester

Harvesting is a process of exchanging metadata with other repositories. As a harvesting *client*, your Dataverse installation can gather metadata records from remote sources. These can be other Dataverse installations or other archives that support OAI-PMH, the standard harvesting protocol. Harvested metadata records will be indexed and made searchable by your users. Clicking on a harvested dataset in the search results takes the user to the original repository. Harvested datasets cannot be edited in your Dataverse installation.

Harvested records can be kept in sync with the original repository through scheduled incremental updates, daily or weekly. Alternatively, harvests can be run on demand, by the Admin.

3.5.2 Managing Harvesting Clients

To start harvesting metadata from a remote OAI repository, you first create and configure a *Harvesting Client*.

Clients are managed on the "Harvesting Clients" page accessible via the *Dashboard*. Click on the *Add Client* button to get started.

The process of creating a new, or editing an existing client, is largely self-explanatory. It is split into logical steps, in a way that allows the user to go back and correct the entries made earlier. The process is interactive and guidance text is provided. For example, the user is required to enter the URL of the remote OAI server. When they click *Next*, the application will try to establish a connection to the server in order to verify that it is working, and to obtain the information about the sets of metadata records and the metadata formats it supports. The choices offered to the user

on the next page will be based on this extra information. If the application fails to establish a connection to the remote archive at the address specified, or if an invalid response is received, the user is given an opportunity to check and correct the URL they entered.

Please note that in some rare cases this GUI may fail to create a client because of some unexpected errors during these real time exchanges with an OAI server that is otherwise known to be valid. For example, in the past we have had issues with servers offering very long lists of sets (*really* long, in the thousands). To allow an admin to still be able to create a client in a situation like that, we provide the REST API that will do so without attempting any validation in real time. This obviously makes it the responsibility of the admin to supply the values that are definitely known to be valid - a working OAI url, the name of a set that does exist on the server, and/or a supported metadata format. See the *Managing Harvesting Clients* section of the *Native API* guide for more information.

"Custom HTTP Header" is part of step 1 of the Create or Edit form. This optional field can be used to configure this client with a specific HTTP header to be added to every OAI request. This is to accommodate a (rare) use case where the remote server may require a special token of some kind in order to offer some content not available to other clients. Most OAI servers offer the same publicly-available content to all clients, so few admins will have a use for this feature. However, it appears in Step 1 of the form screen in case the OAI server requires this token even for the "ListSets" and "ListMetadataFormats" requests, which need to be sent in Step 2 of creating or editing a client. Multiple headers can be supplied separated by \n - actual "backslash" and "n" characters, not a single "new line" character.

Harvesting from Datacite

It is possible to harvest metadata directly from DataCite. Their OAI gateway (https://oai.datacite.org/oai) serves records for every DOI they have registered. Therefore, it is now possible to harvest metadata from any participating institution even if they do not maintain an OAI server of their own. Their OAI implementation offers a concept of a "dynamic set", making it possible to use any query supported by the DataCite search API as though it were a "set". This makes harvesting from them extra flexible, allowing users to harvest virtually any arbitrary subset of metadata records, potentially spanning multiple institutions and registration authorities.

For various reasons, in order to take advantage of this feature harvesting clients must be created via the `/api/harvest/clients` API. Once configured however, harvests can be run from the Harvesting Clients control panel in the UI. See the *Managing Harvesting Clients* section of the *Native API* guide for more information.

How to Stop a Harvesting Run in Progress

Some harvesting jobs, especially the initial full harvest of a very large set - such as the default set of public datasets at IQSS - can take many hours. In case it is necessary to terminate such a long-running job, the following mechanism is provided (note that it is only available to a sysadmin with shell access to the application server): Create an empty file in the domain logs directory with the following name: `stopharvest_<name>.<pid>`, where `<name>` is the nickname of the harvesting client and `<pid>` is the process id of the Application Server (Payara). This flag file needs to be owned by the same user that's running Payara, so that the application can remove it after stopping the job in progress.

For example:

```
sudo touch /usr/local/payara7/glassfish/domains/domain1/logs/stopharvest_bigarchive.70916
sudo chown dataverse /usr/local/payara7/glassfish/domains/domain1/logs/stopharvest_
↪bigarchive.70916
```

Note: If the application server is stopped and restarted, any running harvesting jobs will be killed but may remain marked as in progress in the database. We thus recommend using the mechanism here to stop ongoing harvests prior to a server restart.

What if a Run Fails?

Each harvesting client run logs a separate file per run to the app server's default logging directory (`/usr/local/payara7/glassfish/domains/domain1/logs/` unless you've changed it). Look for filenames in the format `harvest_TARGET_YYYY_MM_DD_timestamp.log` to get a better idea of what's going wrong.

Note that you'll want to run a minimum of Dataverse Software 4.6, optimally 4.18 or beyond, for the best OAI-PMH interoperability.

3.5.3 Harvesting Client Changelog

- As of Dataverse 6.6, it is possible to harvest metadata directly from DataCite. See *Harvesting from Datacite*.
- As of Dataverse 6.6, the publisher value of harvested datasets is now attributed to the dataset's distributor instead of its producer. This change affects all newly harvested datasets. For more information, see https://github.com/IQSS/dataverse/pull/9013
- As of Dataverse 5.13, a new entry called "Custom HTTP Header" has been added to the Step 1. of Create or Edit form. For usage see *Managing Harvesting Clients*.

3.5.4 Harvesting Non-OAI-PMH

DOI2PMH is a community-driven project intended to allow OAI-PMH harvesting from non-OAI-PMH sources.

3.6 Managing Harvesting Server and Sets

Contents:

- *Your Dataverse Installation as an OAI server*
- *How does it work?*
- *OAI Sets*
  - *Important: New Solr schema required!*
- *OAI Set updates*

3.6.1 Your Dataverse Installation as an OAI server

As a harvesting *server*, your Dataverse installation can make some of the local dataset metadata available to remote harvesting clients. These can be other Dataverse installations, or any other clients that support OAI-PMH harvesting protocol. Note that the terms "Harvesting Server" and "OAI Server" are being used interchangeably throughout this guide and in the inline help text.

If you want to learn more about OAI-PMH, you could take a look at DataCite OAI-PMH guide or the OAI-PMH protocol definition.

You might consider adding your OAI-enabled Dataverse installation to this shared list of such instances.

The email portion of *dataverse.mail.system-email* will be visible via OAI-PMH (from the "Identify" verb).

Please note that Dataverse can also harvest metadata from other OAI-PMH servers, like *DataCite*. See *Managing Harvesting Clients* for details.

3.6.2 How does it work?

Only the published datasets in your Dataverse installation can be made harvestable. Remote clients normally keep their records in sync through scheduled incremental updates, daily or weekly, thus minimizing the load on your server. Note that it is only the metadata that are harvested. Remote harvesters will generally not attempt to download the data files associated with the harvested datasets.

Harvesting server can be enabled or disabled on the "Harvesting Server" page accessible via the *Dashboard*. Harvesting server is by default disabled on a brand new, "out of the box" Dataverse installation.

The OAI-PMH endpoint can be accessed at `http(s)://<Your Dataverse Installation FQDN>/oai`. If you want other services to harvest your repository, point them to this URL. *Example URL for 'Identify' verb*: demo.dataverse.org OAI

3.6.3 OAI Sets

Once the service is enabled, you define collections of local datasets that will be available to remote harvesters as *OAI Sets*. Once again, the terms "OAI Set" and "Harvesting Set" are used interchangeably. Sets are defined by search queries. Any such query that finds any number of published, local (non-harvested) datasets can be used to create an OAI set. Sets can overlap local Dataverse collections, and can include as few or as many of your local datasets as you wish. A good way to master the Dataverse Software search query language is to experiment with the Advanced Search page. We also recommend that you consult the *Search API* section of the API Guide.

Once you have entered the search query and clicked *Next*, the number of search results found will be shown on the next screen. This way, if you are seeing a number that's different from what you expected, you can go back and try to re-define the query.

Some useful examples of search queries to define OAI sets:

- A good way to create a set that would include all your local, published datasets is to do so by the Unique Identifier authority registered to your Dataverse installation, for example:

  `dsPersistentId:"doi:1234/"`

  Note that double quotes must be used, since the search field value contains the colon symbol!

  Note also that the search terms limiting the results to published and local datasets **are added to the query automatically**, so you don't need to worry about that.

- A query to create a set to include the datasets from a specific Dataverse collection:

  `parentId:NNN`

  where NNN is the database id of the Dataverse collection object (to verify the database ID, consult the Dataverse table of the SQL database used by the application or use the *View a Dataverse Collection* API endpoint).

  Note that this query does **not** provide datasets that are linked into the specified Dataverse collection.

- A query to create a set to include the datasets from a specific Dataverse collection including datasets that have been deposited into other Dataverse collections but linked into the specified Dataverse collection:

  `subtreePaths:"/NNN"`

  where NNN is the database ID of the Dataverse collection (to verify the database ID, consult the Dataverse table of the SQL database used by the application or use the *View a Dataverse Collection* API endpoint). If the Dataverse collection has one or more parent collections, the subtreePaths query has to include the database IDs of each of the collection's parent collections, up to but excluding the "Root" collection. For example:

`subtreePaths:"/AAA/BBB/NNN"`

where NNN is the database ID of the Dataverse collection containing the desired dataset and AAA and BBB are the database IDs of the parent collections.

- A query to find all the dataset by a certain author:

  `authorName:YYY`

  where YYY is the name.

- Complex queries can be created with multiple logical AND and OR operators. For example,

  `(authorName:YYY OR authorName:ZZZ) AND dsPublicationDate:NNNN`

- Some further query examples:

  For specific datasets using a persistentID:

  `(dsPersistentId:10.5000/ZZYYXX/ OR dsPersistentId:10.5000/XXYYZZ)`

  For all datasets within a specific ID authority:

  `dsPersistentId:10.5000/XXYYZZ`

  For all Dataverse collections with subjects of Astronomy and Astrophysics or Earth and Environmental Sciences:

  `(dvSubject:"Astronomy and Astrophysics" OR dvSubject:"Earth and Environmental Sciences")`

  For all datasets containing the keyword "censorship":

  `keywordValue:censorship`

Important: New Solr schema required!

In order to be able to define OAI sets, your Solr server must be upgraded with the search schema that came with release 4.5 (or later), and all your local datasets must be re-indexed, once the new schema is installed.

3.6.4 OAI Set updates

Every time a new harvesting set is created, or changes are made to an existing set, the contents of the set are automatically updated - the Dataverse installation will find the datasets defined by the query, and attempt to run the metadata export on the ones that haven't been exported yet. Only the datasets for which the export has completed successfully, and the results cached on the filesystem are included in the OAI sets advertised to the harvesting clients!

This is in contrast to how the sets used to be managed in DVN v.3, where sets had to be exported manually before any such changes had effect.

Important: Note however that changes made to the actual dataset metadata do not automatically trigger any corresponding OAI sets to be updated immediately! For example: let's say you have created an OAI set defined by the search query `authorName:king`, that resulted in 43 dataset records. If a new dataset by the same author is added and published, this **does not** immediately add the extra record to the set! It would simply be too expensive, to refresh all the sets every time any changes to the metadata are made.

The OAI set will however be updated automatically by a scheduled metadata export job that runs every night (at 2AM, by default). This export timer is created and activated automatically every time the application is deployed or restarted. See the *Metadata Export* section of the Admin Guide, for more information on the automated metadata exports.

It is still possible however to make changes like this be immediately reflected in the OAI server, by going to the *Harvesting Server* page and clicking the "Run Export" icon next to the desired OAI set.

3.7 Metadata Customization

The Dataverse Software has a flexible data-driven metadata system powered by "metadata blocks" that are listed in the *Appendix* section of the User Guide. In this section we explain the customization options.

Contents:

- *Introduction*
- *About the metadata block TSV*
  - *#metadataBlock properties*
  - *#datasetField (field) properties*
  - *#controlledVocabulary (enumerated) properties*
  - *FieldType definitions*
  - *displayFormat variables*
- *Metadata Block Setup*
  - *Exploring Metadata Blocks*
  - *Setting Up a Dev Environment for Testing*
  - *Editing TSV files*
  - *Loading TSV files into a Dataverse Installation*
  - *#metadataBlock properties*
  - *#datasetField (field) properties*
  - *#controlledVocabulary (enumerated) properties*
  - *Enabling a Metadata Block*
  - *Updating the Solr Schema*
- *Reloading a Metadata Block*
- *Using External Vocabulary Services*
- *Protecting MetadataBlocks*
- *Tips from the Dataverse Community*
- *Development Tasks Specific to Changing Fields in Core Metadata Blocks*
  - *Making a Field Multi-Valued*
- *Footnotes*

3.7.1 Introduction

Before you embark on customizing metadata in your Dataverse installation you should make sure you are aware of the modest amount of customization that is available with your Dataverse installation's web interface. It's possible to hide fields and make fields required or conditionally required by clicking "Edit" at the Dataverse collection level, clicking "General Information" and making adjustments under "Metadata Fields" as described in the *Create a New Dataverse Collection* section of the Dataverse Collection Management page in the User Guide.

Much more customization of metadata is possible, but this is an advanced topic so feedback on what is written below is very welcome. The possibilities for customization include:

- Editing and adding metadata fields
- Editing and adding instructional text (field label tooltips and text box watermarks)
- Editing and adding controlled vocabularies
- Changing which fields depositors must use in order to save datasets (see also *Dataset Templates* section of the User Guide.)
- Changing how saved metadata values are displayed in the UI

Generally speaking it is safer to create your own custom metadata block rather than editing metadata blocks that ship with the Dataverse Software, because changes to these blocks may be made in future releases. If you'd like to make improvements to any of the metadata blocks shipped with the Dataverse Software, please open an issue at https://github.com/IQSS/dataverse/issues so it can be discussed before a pull request is made. Please note that the metadata blocks shipped with the Dataverse Software are based on standards (e.g. DDI for social science) and you can learn more about these standards in the *Appendix* section of the User Guide. If you have developed your own custom metadata block that you think may be of interest to the Dataverse community, please create an issue and consider making a pull request as described in the *Version Control* section of the Developer Guide.

In current versions of the Dataverse Software, custom metadata are no longer defined as individual fields, as they were in Dataverse Network (DVN) 3.x, but in metadata blocks. The Dataverse Software now ships with a citation metadata block, which includes mandatory fields needed for assigning persistent IDs to datasets, and domain specific metadata blocks. For a complete list, see the *Appendix* section of the User Guide.

Definitions of these blocks are loaded into a Dataverse installation in tab-separated value (TSV).[1,2] While it is technically possible to define more than one metadata block in a TSV file, it is good organizational practice to define only one in each file.

The metadata block TSVs shipped with the Dataverse Software are in /scripts/api/data/metadatablocks with the corresponding ResourceBundle property files in /src/main/java/propertyFiles of the Dataverse Software GitHub repo. Human-readable copies are available in this Google Sheets document but they tend to get out of sync with the TSV files, which should be considered authoritative. The Dataverse Software installation process operates on the TSVs, not the Google spreadsheet.

3.7.2 About the metadata block TSV

Here we list and describe the purposes of each section and property of the metadata block TSV.

1. metadataBlock
    - Purpose: Represents the metadata block being defined.
    - Cardinality:
        - 0 or more per Dataverse installation

[1] https://www.iana.org/assignments/media-types/text/tab-separated-values
[2] Although the structure of the data, as you'll see below, violates the "Each record must have the same number of fields" tenet of TSV

- 1 per Metadata Block definition

2. datasetField
   - Purpose: Each entry represents a metadata field to be defined within a metadata block.
   - Cardinality: 1 or more per metadataBlock
3. controlledVocabulary
   - Purpose: Each entry enumerates an allowed value for a given datasetField.
   - Cardinality: zero or more per datasetField

Each of the three main sections own sets of properties:

#metadataBlock properties

Property	Purpose	Allowed values and restrictions
name	A user-definable string used to identify a #metadataBlock	• No spaces or punctuation, except underscore. • By convention, should start with a letter, and use lower camel case[3] • Must not collide with a field of the same name in the same or any other #datasetField definition, including metadata blocks defined elsewhere.[4]
dataverse	If specified, this metadata block will be available only to the Dataverse collection designated here by its alias and to children of that Dataverse collection.	Free text. For an example, see custom_hbgdki.tsv.
displayName	Acts as a brief label for display related to this #metadataBlock.	Should be relatively brief. The limit is 256 character, but very long names might cause display problems.
displayFacet	Label displayed in the search area when this #metadataBlock is configured as a search facet for a collection. See *the API*.	Should be brief. Long names will cause display problems in the search area.
blockURI	Associates the properties in a block with an external URI. Properties will be assigned the global identifier blockURI<name> in the OAI_ORE metadata and archival Bags	The citation #metadataBlock has the blockURI https://dataverse.org/schema/citation/ which assigns a default global URI to terms such as https://dataverse.org/schema/citation/subtitle

[3] https://en.wikipedia.org/wiki/CamelCase
[4] These field names are added to the Solr schema.xml and cannot be duplicated. See "Editing TSV files" for how to check for duplication.

#datasetField (field) properties

Property	Purpose	Allowed values and restrictions	
name	A user-definable string used to identify a #datasetField. Maps directly to field name used by Solr.	• (from DatasetFieldType.java) The internal DDI-like name, no spaces, etc. • (from Solr) Field names should consist of alphanumeric or underscore characters only and not start with a digit. This is not currently strictly enforced, but other field names will not have first class support from all components and back compatibility is not guaranteed. Names with both leading and trailing underscores (e.g. _version_) are reserved. • Must not collide with a field of the same same name in another #metadataBlock definition or any name already included as a field in the Solr index.	
title	Acts as a brief label for display related to this #datasetField.	Should be relatively brief.	
description	Used to provide a description of the field.	Free text	
watermark	A string to initially display in a field as a prompt for what the user should enter.	Free text	
fieldType	Defines the type of content that the field, if not empty, is meant to contain.		• none • date • email • text • textbox • string • url • int • float • See below for fieldtype definitions
displayOrder	Controls the sequence in which the fields are displayed, both for input and presentation.	Non-negative integer.	
displayFormat	Controls how the content is displayed for presentation (not entry). The value of this field may contain one or more special variables (enumerated below). HTML tags, likely in conjunction with one or more of these values, may be used to control the display of content in the web UI.	See below for displayFormat variables	
advancedSearchField	Specify whether this field is available in advanced search.	TRUE (available) or FALSE (not available)	
allowControlled	Specify whether the possible values of this field are determined by values in the #controlledVocabulary section.	TRUE (controlled) or FALSE (not controlled)	

#controlledVocabulary (enumerated) properties

Property	Purpose	Allowed values and restrictions
Datas Field	Specifies the #datasetField to which #datasetField to which this entry applies.	Must reference an existing #datasetField. As a best practice, the value should reference a #datasetField in the current metadata block definition. (It is technically possible to reference an existing #datasetField from another metadata block.)
Value	A short display string, representing an enumerated value for this field. If the identifier property is empty, this value is used as the identifier.	Free text. When defining a boolean, the values "True" and "False" are recommended and "Unknown" can be added if needed.
identifier	A string used to encode the selected enumerated value of a field. If this property is empty, the value of the "Value" field is used as the identifier.	Free text
displayOrde	Control the order in which the enumerated values are displayed for selection. When adding new values, you don't have to add them at the end. You can renumber existing values to update the order in which they appear.	Non-negative integer.

FieldType definitions

Field type	Definition
none	Used for compound fields, in which case the parent field would have no value and display no data entry control.
date	A date, expressed in one of three resolutions of the form YYYY-MM-DD, YYYY-MM, or YYYY.
email	A valid email address. Not indexed for privacy reasons.
text	Any text other than newlines may be entered into this field. The text fieldtype may be used to define a boolean (see "Value" under *#controlledVocabulary (enumerated) properties*).
textbo	Any text may be entered. For input, the Dataverse Software presents a multi-line area that accepts newlines. While any HTML is permitted, only a subset of HTML tags will be rendered in the UI. See the *Supported HTML Tags* section of the Dataset + File Management page in the User Guide.
string	Any text may be entered into this field. The value is stored and indexed exactly as provided, without any text analysis or transformations.
url	If not empty, field must contain a valid URL.
int	An integer value destined for a numeric field.
float	A floating point number destined for a numeric field.

[5] "displayoncreate" was "showabovefold" in Dataverse Software <=4.3.1 (see #3073) but parsing is done based on column order rather than name so this only matters to the person reading the TSV file.

displayFormat variables

These are common ways to use the displayFormat to control how values are displayed in the UI. This list is not exhaustive.

Variable	Description
(blank)	The displayFormat is left blank for primitive fields (e.g. subtitle) and fields that do not take values (e.g. author), since displayFormats do not work for these fields.
#VALUE	The value of the field (instance level).
#NAME	The name of the field (class level).
#EMAIL	For displaying emails.
#VALUE	For displaying the value as a link (if the value entered is a link).
#VALUE	For displaying the value as a link, with the value included in the URL (e.g. if URL is http://emsearch.rutgers.edu/atlas/#VALUE_summary.html, and the value entered is 1001, the field is displayed as 1001 (hyperlinked to http://emsearch.rutgers.edu/atlas/1001_summary.html)).
 	For displaying the image of an entered image URL (used to display images in the producer and distributor logos metadata fields).
#VALUE: - #VALUE: (#VALUE)	Appends and/or prepends characters to the value of the field. e.g. if the displayFormat for the distributorAffiliation is (#VALUE) (wrapped with parens) and the value entered is University of North Carolina, the field is displayed in the UI as (University of North Carolina).
; : ,	Displays the character (e.g. semicolon, comma) between the values of fields within compound fields. For example, if the displayFormat for the compound field "series" is a colon, and if the value entered for seriesName is IMPs and for seriesInformation is A collection of NMR data, the compound field is displayed in the UI as IMPs: A collection of NMR data.

3.7.3 Metadata Block Setup

Now that you understand the TSV format used for metadata blocks, the next step is to attempt to make improvements to existing metadata blocks or create entirely new metadata blocks. For either task, you should have a Dataverse Software development environment set up for testing where you can drop the database frequently while you make edits to TSV files. Once you have tested your TSV files, you should consider making a pull request to contribute your improvement back to the community.

Exploring Metadata Blocks

In addition to studying the TSV files themselves you will probably find the *Metadata Blocks* API helpful in getting a structured dump of metadata blocks in JSON format.

There are also a few older, highly experimental, and subject-to-change API endpoints under the "admin" API documented below but the public API above is preferred.

You can get a dump of metadata fields like this:

```
curl http://localhost:8080/api/admin/datasetfield
```

To see details about an individual field such as "title" in the example below:

```
curl http://localhost:8080/api/admin/datasetfield/title
```

Setting Up a Dev Environment for Testing

You have several options for setting up a dev environment for testing metadata block changes:

- Docker: See *Editing Metadata Blocks* in the Container Guide.
- AWS deployment: See the *Deployment* section of the Developer Guide.
- Full dev environment: See the *Development Environment* section of the Developer Guide.

Editing TSV files

Early in Dataverse Software 4.0 development, metadata blocks were edited in the Google spreadsheet mentioned above and then exported in TSV format. This worked fine when there was only one person editing the Google spreadsheet but now that contributions are coming in from all over, the TSV files are edited directly. We are somewhat painfully aware that another format such as XML might make more sense these days. Please see https://github.com/IQSS/dataverse/issues/4451 for a discussion of non-TSV formats.

Please note that metadata fields share a common namespace so they must be unique. The following curl command will print the list of metadata fields already available in the system:

```
curl http://localhost:8080/api/admin/index/solr/schema
```

We'll use this command again below to update the Solr schema to accomodate metadata fields we've added.

Loading TSV files into a Dataverse Installation

A number of TSV files are loaded into a newly-installed Dataverse installation, becoming the metadata blocks you see in the UI. For the list of metadata blocks that are included with the Dataverse Software out of the box, see the *Appendix* section of the User Guide.

Along with TSV file, there are corresponding ResourceBundle property files with key=value pair here. To add other language files, see the *Configuration* for dataverse.lang.directory JVM Options section, and add a file, for example: "citation_lang.properties" to the path you specified for the `dataverse.lang.directory` JVM option, and then restart the app server.

If you are improving an existing metadata block, the Dataverse Software installation process will load the TSV for you, assuming you edited the TSV file in place. The TSV file for the Citation metadata block, for example, can be found at `scripts/api/data/metadatablocks/citation.tsv`. If any of the below mentioned property values are changed, corresponding ResourceBundle property file has to be edited and stored under `dataverse.lang.directory` location

- name, displayName property under #metadataBlock
- name, title, description, watermark properties under #datasetfield
- DatasetField, Value property under #controlledVocabulary

If you are creating a new custom metadata block (hopefully with the idea of contributing it back to the community if you feel like it would provide value to others), the Dataverse Software installation process won't know about your new TSV file so you must load it manually. The script that loads the TSV files into the system is `scripts/api/setup-datasetfields.sh` and contains a series of curl commands. Here's an example of the necessary curl command with the new custom metadata block in the "/tmp" directory.

```
curl http://localhost:8080/api/admin/datasetfield/load -H "Content-type:  text/
tab-separated-values" -X POST --upload-file /tmp/new-metadata-block.tsv
```

To create a new ResourceBundle, here are the steps to generate key=value pair for the three main sections:

#metadataBlock properties

metadatablock.name=(the value of **name** property from #metadatablock)

metadatablock.displayName=(the value of **displayName** property from #metadatablock)

metadatablock.displayFacet=(the value of **displayFacet** property from #metadatablock)

example:

metadatablock.name=citation

metadatablock.displayName=Citation Metadata

metadatablock.displayFacet=Citation

#datasetField (field) properties

datasetfieldtype.(the value of **name** property from #datasetField).title=(the value of **title** property from #datasetField)

datasetfieldtype.(the value of **name** property from #datasetField).description=(the value of **description** property from #datasetField)

datasetfieldtype.(the value of **name** property from #datasetField).watermark=(the value of **watermark** property from #datasetField)

example:

datasetfieldtype.title.title=Title

datasetfieldtype.title.description=Full title by which the Dataset is known.

datasetfieldtype.title.watermark=Enter title…

#controlledVocabulary (enumerated) properties

controlledvocabulary.(the value of **DatasetField** property from #controlledVocabulary).(the value of **Value** property from #controlledVocabulary)=(the value of **Value** property from #controlledVocabulary)

Since the **Value** property from #controlledVocabulary is free text, while creating the key, it has to be converted to lowercase, replace space with underscore, and strip accents.

example:

controlledvocabulary.subject.agricultural_sciences=Agricultural Sciences

controlledvocabulary.language.marathi_(marathi)=Marathi (Maru0101u1E6Dhu012B)

Enabling a Metadata Block

Running a curl command like "load" example above should make the new custom metadata block available within the system but in order to start using the fields you must either enable it from the UI (see *General Information* section of Dataverse Collection Management in the User Guide) or by running a curl command like the one below using a superuser API token. In the example below we are enabling the "journal" and "geospatial" metadata blocks for the root Dataverse collection:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X POST -H "Content-type:application/json" -d "[\
"journal\",\"geospatial\"]" http://localhost:8080/api/dataverses/:root/metadatablocks
```

Updating the Solr Schema

Once you have enabled a new metadata block you should be able to see the new fields in the GUI but before you can save the dataset, you must add additional fields to your Solr schema.

An API endpoint of your Dataverse installation provides you with a generated set of all fields that need to be added to the Solr schema configuration, including any enabled metadata schemas:

```
curl "http://localhost:8080/api/admin/index/solr/schema"
```

You can use `update-fields.sh` to easily add these to the Solr schema you installed for your Dataverse installation.

The script needs a target XML file containing your Solr schema. (See the *Prerequisites* section of the Installation Guide for a suggested location on disk for the Solr schema file.)

You can either pipe the downloaded schema to the script or provide the file as an argument. (We recommended you to take a look at usage output of `update-fields.sh -h`)

Listing 1: Example usage of `update-fields.sh`

```
curl "http://localhost:8080/api/admin/index/solr/schema" | update-fields.sh /usr/local/
↪solr/server/solr/collection1/conf/schema.xml
```

You will need to reload your Solr schema via an HTTP-API call, targeting your Solr instance:

```
curl "http://localhost:8983/solr/admin/cores?action=RELOAD&core=collection1"
```

You can easily roll your own little script to automate the process (which might involve fetching the schema bits from some place else than your Dataverse installation). For your convenience, an example script `solr-driver.sh` has been added.

Please note that reconfigurations of your Solr index might require a re-index. Usually release notes indicate a necessary re-index, but for your custom metadata you will need to keep track on your own.

Please note also that if you are going to make a pull request updating `conf/solr/schema.xml` with fields you have added, you should first load all the custom metadata blocks in `scripts/api/data/metadatablocks` (including ones you don't care about) to create a complete list of fields. (This might change in the future.) Please see *Updating the Solr Schema (Developers)* in the Developer Guide.

3.7.4 Reloading a Metadata Block

As mentioned above, changes to metadata blocks that ship with the Dataverse Software will be made over time to improve them and release notes will sometimes instruct you to reload an existing metadata block. The syntax for reloading is the same as loading. Here's an example with the "citation" metadata block:

```
curl http://localhost:8080/api/admin/datasetfield/load -H "Content-type:  text/
tab-separated-values" -X POST --upload-file citation.tsv
```

Great care must be taken when reloading a metadata block. Matching is done on field names (or identifiers and then names in the case of controlled vocabulary values) so it's easy to accidentally create duplicate fields.

The ability to reload metadata blocks means that SQL update scripts don't need to be written for these changes. See also the *SQL Upgrade Scripts* section of the Developer Guide.

3.7.5 Using External Vocabulary Services

The Dataverse software has a mechanism to associate specific fields defined in metadata blocks with a vocabulary(ies) managed by external services. The mechanism relies on trusted third-party Javascripts. The mapping from field type to external vocabulary(ies) is managed via the *:CVocConf* setting.

This functionality may require significant effort to configure and is likely to evolve in subsequent Dataverse software releases.

The effect of configuring this mechanism is similar to that of defining a field in a metadata block with 'allowControlledVocabulary=true':

- Users are able to select from a controlled list of values.
- Values can be shown in any language the term has been defined in.

In general, the external vocabulary support mechanism may be a better choice for large vocabularies, hierarchical/structured vocabularies, and/or vocabularies managed by third-parties. In addition, the external vocabulary mechanism differs from the internal controlled vocabulary mechanism in several ways that may make it a preferred option:

- the machine-readable URI form of a vocabulary is stored in the Dataverse database and can be included in exported metadata files.
- vocabulary mappings can be changed without changing the metadata block, making it possible for different Dataverse installations to use different vocabularies in the same field.
- mappings can associate a field with more than one vocabulary.
- mappings can be configured to also allow custom/free-text entries as well as vocabulary values.
- mappings can be configured for compound fields and a user's selection of a given vocabulary value can be used to fill in related child fields (e.g. selection of a keyword could fill in a vocabulary name field as well).
- removing a mapping does not affect stored values (the field would revert to allowing free text).

The specifics of the user interface for entering/selecting a vocabulary term and how that term is then displayed are managed by third-party Javascripts. The initial Javascripts that have been created provide auto-completion, displaying a list of choices that match what the user has typed so far, but other interfaces, such as displaying a tree of options for a hierarchical vocabulary, are possible. Similarly, existing scripts do relatively simple things for displaying a term - showing the term's name in the appropriate language and providing a link to an external URL with more information, but more sophisticated displays are possible.

Scripts supporting use of vocabularies from services supporting the SKOSMOS protocol (see https://skosmos.org), retrieving ORCIDs (from https://orcid.org), services based on Ontoportal product (see https://ontoportal.org/), and using ROR (https://ror.org/) are available https://github.com/gdcc/dataverse-external-vocab-support. (Custom scripts can also be used and community members are encouraged to share new scripts through the dataverse-external-vocab-support repository.)

Configuration involves specifying which fields are to be mapped, to which Solr field they should be indexed, whether free-text entries are allowed, which vocabulary(ies) should be used, what languages those vocabulary(ies) are available in, and several service protocol and service instance specific parameters, including the ability to send HTTP headers on calls to the service. These are all defined in the *:CVocConf* setting as a JSON array. Details about the required elements as well as example JSON arrays are available at https://github.com/gdcc/dataverse-external-vocab-support, along with an example metadata block that can be used for testing. The scripts required can be hosted locally or retrieved dynamically from https://gdcc.github.io/ (similar to how dataverse-previewers work).

Since external vocabulary scripts can change how fields are indexed (storing an identifier and name and/or values in different languages), updating the Solr schema as described in *Updating the Solr Schema* should be done after adding new scripts to your configuration.

Please note that in addition to the *:CVocConf* described above, an alternative is the *:ControlledVocabularyCustomJavaScript* setting.

3.7.6 Protecting MetadataBlocks

Dataverse can be configured to only allow entries for a metadata block to be changed (created, edited, deleted) by entities that know a defined secret key. Metadata blocks protected by such a key are referred to as "System" metadata blocks. A primary use case for system metadata blocks is to handle metadata created by third-party tools interacting with Dataverse where unintended changes to the metadata could cause a failure. Examples might include archiving systems or workflow engines. To protect an existing metadatablock, one must set a key (recommended to be long and un-guessable) for that block:

dataverse.metadata.block-system-metadata-keys.<block name>=<key value>

This can be done using system properties (see *JVM Options*), environment variables or other MicroProfile Config mechanisms supported by the app server.

See Payara docs for supported sources. Note that a Payara restart may be required to enable the new option.

For these secret keys, Payara password aliases are recommended.

Alias creation example using the codemeta metadata block (actual name: codeMeta20):

```
echo "AS_ADMIN_ALIASPASSWORD=1234ChangeMeToSomethingLong" > /tmp/key.txt
asadmin create-password-alias --passwordfile /tmp/key.txt dataverse.metadata.
↪block-system-metadata-keys.codeMeta20
rm /tmp/key.txt
```

Alias deletion example for the codemeta metadata block (removes protected status)

```
asadmin delete-password-alias dataverse.metadata.block-system-metadata-keys.
↪codeMeta20
```

A Payara restart is required after these example commands.

When protected via a key, a metadata block will not be shown in the user interface when a dataset is being created or when metadata is being edited. Entries in such a system metadata block will be shown to users, consistent with Dataverse's design in which all metadata in published datasets is publicly visible.

Note that protecting a block with required fields, or using a template with an entry in a protected block, will make it impossible to create a new dataset via the user interface. Also note that for this reason protecting the citation metadatablock is not recommended. (Creating a dataset also automatically sets the date of deposit field in the citation block, which would be prohibited if the citation block is protected.)

To remove proted status and return a block to working normally, remove the associated key.

To add metadata to a system metadata block via API, one must include an additional key of the form

mdkey.<blockName>=<key value>

as an HTTP Header or query parameter (case sensitive) for each system metadata block to any API call in which metadata values are changed in that block. Multiple keys are allowed if more than one system metadatablock is being changed in a given API call.

For example, following the *Add Dataset Metadata* example from the *Dataset Semantic Metadata API*:

```
curl -X PUT -H X-Dataverse-key:$API_TOKEN -H 'Content-Type: application/ld+json' -H
↪'mdkey.codeMeta20:1234ChangeMeToSomethingLong' -d '{"codeVersion": "1.0.0", "@context":
↪{"codeVersion": "https://schema.org/softwareVersion"}}' "$SERVER_URL/api/datasets/
↪$DATASET_ID/metadata"

curl -X PUT -H X-Dataverse-key:$API_TOKEN -H 'Content-Type: application/ld+json' -d '{
↪"codeVersion": "1.0.1", "@context":{"codeVersion": "https://schema.org/softwareVersion
```

(continues on next page)

(continued from previous page)

```
↪"}}' "$SERVER_URL/api/datasets/$DATASET_ID/metadata?mdkey.
↪codeMeta20=1234ChangeMeToSomethingLong&replace=true"
```

3.7.7 Tips from the Dataverse Community

When creating new metadata blocks, please review the *Text* section of the Style Guide, which includes guidance about naming metadata fields and writing text for metadata tooltips and watermarks.

If there are tips that you feel are omitted from this document, please open an issue at https://github.com/IQSS/dataverse/issues and consider making a pull request to make improvements. You can find this document at https://github.com/IQSS/dataverse/blob/develop/doc/sphinx-guides/source/admin/metadatacustomization.rst

3.7.8 Development Tasks Specific to Changing Fields in Core Metadata Blocks

When it comes to the fields from the core blocks that are distributed with Dataverse (such as Citation, Social Science and Geospatial blocks), code dependencies may exist in Dataverse, primarily in the Import and Export subsystems, on these fields being configured a certain way. So, if it becomes necessary to modify one of such core fields, code changes may be necessary to accompany the change in the block tsv, plus some sample and test files maintained in the Dataverse source tree will need to be adjusted accordingly.

Making a Field Multi-Valued

As a recent real life example, a few fields from the Citation and Social Science block were changed to support multiple values, in order to accommodate specific needs of some community member institutions. A PR for one of these fields, `alternativeTitle` from the Citation block is linked below. Each time a number of code changes, plus some changes in the sample metadata files in the Dataverse code tree had to be made. The checklist below is to help another developer in the event that a similar change becomes necessary in the future. Note that some of the steps below may not apply 1:1 to a different metadata field, depending on how it is exported and imported in various formats by Dataverse. It may help to consult the PR #9440 as a specific example of the changes that had to be made for the `alternativeTitle` field.

- Change the value from `FALSE` to `TRUE` in the `allowmultiples` column of the .tsv file for the block.
- Change the value of the `multiValued` attribute for the search field in the Solr schema (`conf/solr/x.x.x/schema.xml`).
- Modify the DDI import code (`ImportDDIServiceBean.java`) to support multiple values. (You may be able to use the change in the PR above as a model.)
- Modify the DDI export utility (`DdiExportUtil.java`).
- Modify the OpenAire export utility (`OpenAireExportUtil.java`).
- Modify the following JSON source files in the Dataverse code tree to actually include multiple values for the field (two should be quite enough!): `scripts/api/data/dataset-create-new-all-default-fields.json`, `src/test/java/edu/harvard/iq/dataverse/export/dataset-all-defaults.txt`, `src/test/java/edu/harvard/iq/dataverse/export/ddi/dataset-finch1.json` and `src/test/java/edu/harvard/iq/dataverse/export/ddi/dataset-create-new-all-ddi-fields.json`. (These are used as examples for populating datasets via the import API and by the automated import and export code tests).
- Similarly modify the following XML files that are used by the DDI export code tests: `src/test/java/edu/harvard/iq/dataverse/export/ddi/dataset-finch1.xml` and `src/test/java/edu/harvard/iq/dataverse/export/ddi/exportfull.xml`.

- Make sure all the automated unit and integration tests are passing. See *Testing* in the Developer Guide.
- Write a short release note to announce the change in the upcoming release. See *Writing a Release Note Snippet* in the Developer Guide.
- Make a pull request.

3.7.9 Footnotes

3.8 Metadata Export

Contents:

- *Automatic Exports*
- *Batch Exports Through the API*
- *Export Failures*
- *Downloading Metadata via GUI*
- *Downloading Metadata via API*
- *Exporter Configuration*

3.8.1 Automatic Exports

Publishing a dataset automatically starts a metadata export job, that will run in the background, asynchronously. Once completed, it will make the dataset metadata exported and cached in all the supported formats listed under *Supported Metadata Export Formats* in the *Dataset + File Management* section of the User Guide.

A scheduled timer job that runs nightly will attempt to export any published datasets that for whatever reason haven't been exported yet. This timer is activated automatically on the deployment, or restart, of the application. So, again, no need to start or configure it manually. (See the *Dataverse Installation Application Timers* section of this Admin Guide for more information.)

3.8.2 Batch Exports Through the API

In addition to the automated exports, a Dataverse installation admin can start a batch job through the API. The following four API calls are provided:

```
curl http://localhost:8080/api/admin/metadata/exportAll
```

```
curl http://localhost:8080/api/admin/metadata/reExportAll
```

```
curl http://localhost:8080/api/admin/metadata/reExportAll?olderThan=<YYYY-MM-DD>
```

```
curl http://localhost:8080/api/admin/metadata/reExportAll?formats=Datacite,croissant
```

```
curl http://localhost:8080/api/admin/metadata/clearExportTimestamps
```

```
curl http://localhost:8080/api/admin/metadata/:persistentId/reExportDataset?
persistentId=doi:10.5072/FK2/AAA000
```

The first will attempt to export all the published, local (non-harvested) datasets that haven't been exported yet. The second will *force* a re-export of every published, local dataset, regardless of whether it has already been exported or not. With the optional olderThan query parameter, the second will *force* re-export of all published, local datasets that were last exported before the olderThan date. Another optional parameter, `formats` (new as of v6.12) allows an administrator to force a reexport of specific metadata formats only.

The first two calls return a status message informing the administrator that the process has been launched (`{"status":"WORKFLOW_IN_PROGRESS"}`). The administrator can check the progress of the process via log files: `[Payara directory]/glassfish/domains/domain1/logs/export_[time stamp].log`.

Instead of running "reExportAll" the same can be accomplished using "clearExportTimestamps" followed by "exportAll". The difference is that when exporting prematurely fails due to some problem, the datasets that did not get exported yet still have the timestamps cleared. A next call to exportAll will skip the datasets already exported and try to export the ones that still need it. Calling clearExportTimestamps should return `{"status":"OK", "data":{"message":"cleared: X"}}` where "X" is the total number of datasets cleared.

The reExportDataset call gives you the opportunity to *force* a re-export of only a specific dataset and (with some script automation) could allow you the export specific batches of datasets. This might be usefull when handling exporting problems or when reExportAll takes too much time and is overkill. Note that *Export Metadata of a Dataset in Various Formats* is a related API.

reExportDataset can be called with either `persistentId` (as shown above, with a DOI) or with the database id of a dataset (as shown below, with "42" as the database id).

```
curl http://localhost:8080/api/admin/metadata/42/reExportDataset
```

An optional parameter, `formats` (new as of v6.12) can be used to force a reexport of specific metadata formats only. For example:

```
curl http://localhost:8080/api/admin/metadata/42/reExportDataset?formats=oai_dc,oai_ddi
```

Note, that creating, modifying, or re-exporting an OAI set will also attempt to export all the unexported datasets found in the set.

3.8.3 Export Failures

An export batch job, whether started via the API, or by the application timer, will leave a detailed log in your configured logs directory. This is the same location where your main app server logs are found. The name of the log file is `export_[timestamp].log` - for example, *export_2016-08-23T03-35-23.log*. The log will contain the numbers of datasets processed successfully and those for which metadata export failed, with some information on the failures detected. Please attach this log file if you need to contact Dataverse Project support about metadata export problems.

3.8.4 Downloading Metadata via GUI

The *Dataset + File Management* section of the User Guide explains how end users can download the metadata formats above from your Dataverse installation's GUI.

3.8.5 Downloading Metadata via API

The *Native API* section of the API Guide explains how end users can download the metadata formats above via API.

3.8.6 Exporter Configuration

Two exporters - Schema.org JSONLD and OpenAire - use an algorithm to determine whether an author, or contact, name belongs to a person or organization. While the algorithm works well, there are cases in which it makes mistakes, usually inferring that an organization is a person.

The Dataverse software implements two jvm-options that can be used to tune the algorithm:

- *dataverse.person-or-org.assume-comma-in-person-name* - boolean, default false. If true, Dataverse will assume any name without a comma must be an organization. This may be most useful for curated Dataverse instances that enforce the "family name, given name" convention.
- *dataverse.person-or-org.org-phrase-array* - a JsonArray of strings. Any name that contains one of the strings is assumed to be an organization. For example, "Project" is a word that is not otherwise associated with being an organization.

3.9 Dataverse Installation Application Timers

Your Dataverse installation uses timers to automatically run scheduled Harvest and Metadata export jobs.

Contents:

- *Dedicated timer server in a Dataverse Installation server cluster*
- *Harvesting Timers*
- *Metadata Export Timer*
- *Saved Searches Links Timer*
- *Known Issues*

3.9.1 Dedicated timer server in a Dataverse Installation server cluster

When running a Dataverse installation cluster - i.e. multiple Dataverse installation application servers talking to the same database - **only one** of them must act as the *dedicated timer server*. This is to avoid starting conflicting batch jobs on multiple nodes at the same time.

This does not affect a single-server installation. So you can safely skip this section unless you are running a multi-server cluster.

The following JVM option instructs the application to act as the dedicated timer server:

```
-Ddataverse.timerServer=true
```

IMPORTANT: Note that this option is automatically set by the Dataverse Software installer script. That means that when **configuring a multi-server cluster**, it will be the responsibility of the installer to remove the option from the domain.xml of every node except the one intended to be the timer server.

3.9.2 Harvesting Timers

These timers are created when scheduled harvesting is enabled by a local admin user (via the "Manage Harvesting Clients" page).

In a multi-node cluster, all these timers will be created on the dedicated timer node (and not necessarily on the node where the harvesting clients were created and/or saved).

A timer will be automatically removed when a harvesting client with an active schedule is deleted, or if the schedule is turned off for an existing client.

3.9.3 Metadata Export Timer

This timer is created automatically whenever the application is deployed or restarted. There is no admin user-accessible configuration for this timer.

This timer runs a daily job that tries to export all the local, published datasets that haven't been exported yet, in all supported metadata formats, and cache the results on the filesystem. (Note that normally an export will happen automatically whenever a dataset is published. This scheduled job is there to catch any datasets for which that export did not succeed, for one reason or another). Also, since this functionality has been added in Dataverse Software 4.5: if you are upgrading from a previous version, none of your datasets are exported yet. So the first time this job runs, it will attempt to export them all.

This daily job will also update all the harvestable OAI sets configured on your server, adding new and/or newly published datasets or marking deaccessioned datasets as "deleted" in the corresponding sets as needed.

This job is automatically scheduled to run at 2AM local time every night.

3.9.4 Saved Searches Links Timer

This timer is created automatically from an @Schedule annotation on the makeLinksForAllSavedSearchesTimer method of the SavedSearchServiceBean when the bean is deployed.

This timer runs a weekly job to create links for any saved searches that haven't been linked yet.

This job is automatically scheduled to run once a week at 12:30AM local time on Sunday. If really necessary, it is possible to change that time by deploying the application war file with an ejb-jar.xml file in the WEB-INF directory of the war file. A `sample file` would run the job every Tuesday at 2:30PM. The schedule can be modified to your choice by editing the fields in the session section. If other EJBs require some form of configuration using an ejb-jar file, there should be one ejb-jar file for the entire application, which can have different sections for each EJB. Below are instructions for the simple case of adding the ejb-jar.xml for the first time and making a custom schedule for the saved search timer.

- Create or edit dataverse/src/main/webapp/WEB-INF/ejb-jar.xml, following the `sample file` provided.
- Edit the parameters in the <schedule> section of the ejb-jar file in the WEB-INF directory to suit your preferred schedule
  - The provided parameters in the sample file are <minute>, <hour>, and <dayOfWeek>; additional parameters are available
    - For a complete reference for calendar expressions that can be used to schedule Timer services see: https://docs.oracle.com/javaee/7/tutorial/ejb-basicexamples004.htm
- Build and deploy the application
- Alternatively, you can insert an ejb-jar.xml file into a provided Dataverse Software war file without building the application.

- Check if there is already an ejb-jar.xml file in the war file
  - jar tvf $DATAVERSE-WAR-FILENAME | grep ejb-jar.xml
    - if the response includes " WEB-INF/ejb-jar.xml", you will need to extract the ejb-jar.xml file for editing
    - jar xvf $DATAVERSE-WAR-FILENAME WEB-INF/ejb-jar.xml
    - edit the extracted WEB-INF/ejb-jar.xml, following the `sample file` provided.
    - if the response is empty, create a WEB-INF directory and create en ejb-jar.xml file in it, following the `sample file` provided.
    - edit the parameters in the <schedule> section of the WEB-INF/ejb-jar.xml to suit your preferred schedule
- Insert the edited WEB-INF/ejb-jar.xml into the dataverse war file
  - jar uvf $DATAVERSE-WAR-FILENAME WEB-INF/ejb-jar.xml
- Deploy the war file

See also *Saved Search* in the API Guide.

3.9.5 Known Issues

We've received several reports of an intermittent issue where the application fails to deploy with the error message "EJB Timer Service is not available." Please see the *Troubleshooting* section of this guide for a workaround.

3.10 Make Data Count

Make Data Count is a project to collect and standardize metrics on data use, especially views, downloads, and citations. The Dataverse Software can integrate Make Data Count to collect and display usage metrics including counts of dataset views, file downloads, and dataset citations.

Contents:

- *Introduction*
- *Architecture*
- *Limitations for Dataverse Installations Using Handles Rather Than DOIs*
- *Configuring Your Dataverse Installation for Make Data Count Views and Downloads*
  - *Enable Logging for Make Data Count*
  - *Enable or Disable Display of Make Data Count Metrics*
  - *Configure Counter Processor*
  - *Populate Views and Downloads for the First Time*
  - *Populate Views and Downloads Nightly*
  - *Sending Usage Metrics to the DataCite Hub*
- *Configuring Your Dataverse Installation for Make Data Count Citations*

- *Retrieving Make Data Count Metrics from the DataCite Hub*
- *Retrieving Make Data Count Metrics from a Dataverse Installation*

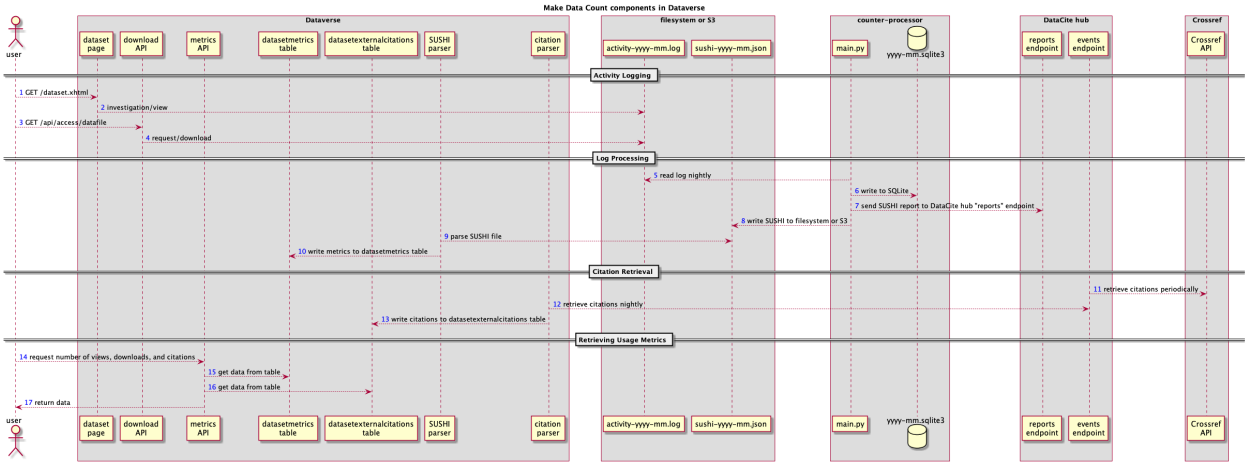
3.10.1 Introduction

Make Data Count is part of a broader Research Data Alliance (RDA) Data Usage Metrics Working Group which helped to produce a specification called the COUNTER Code of Practice for Research Data (PDF, HTML) that the Dataverse Software makes every effort to comply with. The Code of Practice (CoP) is built on top of existing standards such as COUNTER and SUSHI that come out of the article publishing world. The Make Data Count project has emphasized that they would like feedback on the code of practice. You can keep up to date on the Make Data Count project by subscribing to their newsletter.

3.10.2 Architecture

Dataverse installations who would like support for Make Data Count must install Counter Processor, a Python project created by California Digital Library (CDL) which is part of the Make Data Count project and which runs the software in production as part of their DASH data sharing platform.

The diagram below shows how Counter Processor interacts with your Dataverse installation and the DataCite hub, once configured. Dataverse installations using Handles rather than DOIs should note the limitations in the next section of this page.



The most important takeaways from the diagram are:

- Once enabled, your Dataverse installation will log activity (views and downloads) to a specialized date-stamped file.
- You should run Counter Processor once a day to create reports in SUSHI (JSON) format that are saved to disk for your Dataverse installation to process and that are sent to the DataCite hub.
- You should set up a cron job to have your Dataverse installation process the daily SUSHI reports, updating the Dataverse installation database with the latest metrics.
- You should set up a cron job to have your Dataverse installation pull the latest list of citations for each dataset on a periodic basis, perhaps weekly or daily. These citations come from Crossref via the DataCite hub.
- APIs are available in the Dataverse Software to retrieve Make Data Count metrics: views, downloads, and citations.

3.10.3 Limitations for Dataverse Installations Using Handles Rather Than DOIs

Data repositories using Handles and other identifiers are not supported by Make Data Count but in the notes following a July 2018 webinar, you can see the Make Data Count project's response on this topic. In short, the DataCite hub does not want to receive reports for non-DOI datasets. Additionally, citations are only available from the DataCite hub for datasets that have DOIs. See also the table below.

	DOIs	Handles
Out of the box	Classic download counts	Classic download counts
Make Data Count	MDC views, MDC downloads, MDC citations	MDC views, MDC downloads

This being said, the Dataverse Software usage logging can still generate logs and process those logs with Counter Processor to create json that details usage on a dataset level. Dataverse installations can ingest this locally generated json.

When editing the `counter-processor-config.yaml` file mentioned below, make sure that the `upload_to_hub` boolean is set to `False`.

3.10.4 Configuring Your Dataverse Installation for Make Data Count Views and Downloads

If you haven't already, follow the steps for installing Counter Processor in the *Prerequisites* section of the Installation Guide.

Enable Logging for Make Data Count

To make your Dataverse installation log dataset usage (views and downloads) for Make Data Count, you must set the `:MDCLogPath` database setting. See *:MDCLogPath* for details.

If you wish to start logging in advance of setting up other components, or wish to log without display MDC metrics for any other reason, you can set the optional `:DisplayMDCMetrics` database setting to false. See *:DisplayMDCMetrics* for details.

After you have your first day of logs, you can process them the next day.

Enable or Disable Display of Make Data Count Metrics

By default, when MDC logging is enabled (when `:MDCLogPath` is set), your Dataverse installation will display MDC metrics instead of it's internal (legacy) metrics. You can avoid this (e.g. to collect MDC metrics for some period of time before starting to display them) by setting `:DisplayMDCMetrics` to false.

The following discussion assumes `:MDCLogPath` has been set to `/usr/local/payara7/glassfish/domains/domain1/logs/mdc` You can also decide to display MDC metrics along with Dataverse's traditional download counts from the time before MDC was enabled. To do this, set the *:MDCStartDate* to when you started MDC logging.

Configure Counter Processor

- First, become the "counter" Unix user.
  - `sudo su - counter`
- Change to the directory where you installed Counter Processor.
  - `cd /usr/local/counter-processor-1.06`
- Download `counter-processor-config.yaml` to `/usr/local/counter-processor-1.06`.
- Edit the config file and pay particular attention to the FIXME lines.
  - `vim counter-processor-config.yaml`

Populate Views and Downloads for the First Time

Soon we will be setting up a cron job to run nightly but we start with a single successful configuration and manual run of Counter Processor and calls to your Dataverse installation's APIs. (The scripts discussed in the next section automate the steps described here, including creating empty log files if you're starting mid-month.)

- Change to the directory where you installed Counter Processor.
  - `cd /usr/local/counter-processor-1.06`
- If you are running Counter Processor for the first time in the middle of a month, you will need create blank log files for the previous days. e.g.:
  - `cd /usr/local/payara7/glassfish/domains/domain1/logs/mdc`
  - `touch counter_2019-02-01.log`
  - `...`
  - `touch counter_2019-02-20.log`
- Run Counter Processor.
  - `CONFIG_FILE=counter-processor-config.yaml python39 main.py`
  - A JSON file in SUSHI format will be created in the directory you specified under "output_file" in the config file.
- Populate views and downloads for your datasets based on the SUSHI JSON file. The "/tmp" directory is used in the example below.
  - `curl -X POST "http://localhost:8080/api/admin/makeDataCount/addUsageMetricsFromSushiReport?reportOnDisk=/tmp/make-data-count-report.json"`
- Verify that views and downloads are available via API.
  - Now that views and downloads have been recorded in the Dataverse installation's database, you should make sure you can retrieve them from a dataset or two. Use the *Dataset Metrics* endpoints in the *Native API* section of the API Guide.

Populate Views and Downloads Nightly

Running `main.py` to create the SUSHI JSON file and the subsequent calling of the Dataverse Software API to process it should be added as a cron job.

The Dataverse Software provides an example script that run the steps to process new accesses and uploads and update your Dataverse installation's database `counter_daily.sh` The script should be configured for your environment and can be run manually or as a cron job.

Sending Usage Metrics to the DataCite Hub

Once you are satisfied with your testing, you should contact support@datacite.org for your JSON Web Token and change "upload_to_hub" to "True" in the config file. The next time you run `main.py` the following metrics will be sent to the DataCite hub for each published dataset:

- Views ("investigations" in COUNTER)
- Downloads ("requests" in COUNTER)

3.10.5 Configuring Your Dataverse Installation for Make Data Count Citations

Please note: as explained in the note above about limitations, this feature is not available to Dataverse installations that use Handles.

To configure your Dataverse installation to pull citations from the test vs. production DataCite server see *Legacy Single PID Provider: dataverse.pid.datacite.rest-api-url* in the Installation Guide.

Please note that in the curl example, Bash environment variables are used with the idea that you can set a few environment variables and copy and paste the examples as is. For example, "$DOI" could become "doi:10.5072/FK2/BL2IBM" by issuing the following export command from Bash:

```
export DOI="doi:10.5072/FK2/BL2IBM"
```

To confirm that the environment variable was set properly, you can use echo like this:

```
echo $DOI
```

On some periodic basis (perhaps weekly) you should call the following curl command for each published dataset to update the list of citations that have been made for that dataset. The example `counter_weekly.sh` will do this for you. The script should be configured for your environment and can be run manually or as a cron job.

```
curl -X POST "http://localhost:8080/api/admin/makeDataCount/:persistentId/
updateCitationsForDataset?persistentId=$DOI"
```

Citations will be retrieved for each published dataset and recorded in the your Dataverse installation's database.

Note that the *dataverse.api.mdc.min-delay-ms* setting can be used to avoid getting rate-limit errors from DataCite.

For how to get the citations out of your Dataverse installation, see "Retrieving Citations for a Dataset" under *Dataset Metrics* in the *Native API* section of the API Guide.

Please note that while the Dataverse Software has a metadata field for "Related Dataset" this information is not currently sent as a citation to Crossref.

3.10.6 Retrieving Make Data Count Metrics from the DataCite Hub

The following metrics can be downloaded directly from the DataCite hub (see https://support.datacite.org/docs/eventdata-guide) for datasets hosted by Dataverse installations that have been configured to send these metrics to the hub:

- Total Views for a Dataset
- Unique Views for a Dataset
- Total Downloads for a Dataset
- Downloads for a Dataset
- Citations for a Dataset (via Crossref)

3.10.7 Retrieving Make Data Count Metrics from a Dataverse Installation

The Dataverse Software API endpoints for retrieving Make Data Count metrics are described below under *Dataset Metrics* in the *Native API* section of the API Guide.

Please note that it is also possible to retrieve metrics from the DataCite hub itself via https://api.datacite.org

3.11 Integrations

Now that you've installed a Dataverse installation, you might want to set up some integrations with other systems. Many of these integrations are open source and are cross listed in the *Apps* section of the API Guide.

Contents:

- *Getting Data In*
  - *GitHub*
  - *Dropbox*
  - *Open Science Framework (OSF)*
  - *RSpace*
  - *Open Journal Systems (OJS) and OPS*
  - *Amnesia*
  - *SampleDB*
  - *RedCap*
  - *GitLab*
  - *iRODS*
  - *Integrations Dashboard*
  - *Globus*
  - *DataLad*
  - *Open OnDemand*

    - *Galaxy*
- *Embedding Data on Websites*
    - *OpenScholar*
- *Analysis and Computation*
    - *Data Explorer*
    - *Compute Button*
    - *Binder*
    - *Renku*
    - *Avgidea Data Search*
    - *JupyterHub*
    - *Rclone*
    - *Open OnDemand*
- *Discoverability*
    - *Geodisy*
    - *DataONE*
- *Research Data Preservation*
    - *Archivematica*
    - *RDA BagIt (BagPack) Archiving*
- *Future Integrations*

3.11.1 Getting Data In

A variety of integrations are oriented toward making it easier for your researchers to deposit data into your Dataverse installation.

GitHub

GitHub can be integrated with a Dataverse installation in multiple ways.

One Dataverse integration is implemented via a Dataverse Uploader GitHub Action. It is a reusable, composite workflow for uploading a git repository or subdirectory into a dataset on a target Dataverse installation. The action is customizable, allowing users to choose to replace a dataset, add to the dataset, publish it or leave it as a draft version in the Dataverse installation. The action provides some metadata to the dataset, such as the origin GitHub repository, and it preserves the directory tree structure.

For instructions on using Dataverse Uploader GitHub Action, visit https://github.com/marketplace/actions/dataverse-uploader-action

In addition to the Dataverse Uploader GitHub Action, the *Integrations Dashboard* also enables a pull of data from GitHub to a dataset.

Dropbox

If your researchers have data on Dropbox, you can make it easier for them to get it into your Dataverse installation by setting the *dataverse.dropbox.key* JVM option described in the *Configuration* section of the Installation Guide.

Open Science Framework (OSF)

The Center for Open Science's Open Science Framework (OSF) is an open source software project that facilitates open collaboration in science research across the lifespan of a scientific project.

OSF can be integrated with a Dataverse installation in multiple ways.

Researcher can configure OSF itself to deposit to your Dataverse installation by following instructions from OSF.

In addition to the method mentioned above, the *Integrations Dashboard* also enables a pull of data from OSF to a dataset.

RSpace

RSpace is an affordable and secure enterprise grade electronic lab notebook (ELN) for researchers to capture and organize data.

For instructions on depositing data from RSpace to your Dataverse installation, your researchers can visit https://www.researchspace.com/help-and-support-resources/dataverse-integration/

As shown in a video, a Data Management Plan (DPM) can be added into RSpace and the research records and associated data can then be sent to Dataverse. Dataverse generates a Persistent Identifier (PID, often a DOI) for the dataset, and RSpace automatically puts the PID link under "Research Outputs" in the DPM.

Open Journal Systems (OJS) and OPS

Open Journal Systems (OJS) is a journal management and publishing system that has been developed by the Public Knowledge Project to expand and improve access to research.

Open Preprint Systems (OPS) is similar, but for preprints.

The following plugin adds data sharing and preservation to the OJS/OPS publication process.

OJS/OPS 3.x can use https://github.com/lepidus/dataversePlugin

OJS 2.x can use https://github.com/pkp/ojs/tree/ojs-2_4_8-5/plugins/generic/dataverse

Amnesia

Amnesia is a flexible data anonymization tool that transforms relational and transactional databases to datasets where formal privacy guarantees hold. Amnesia transforms original data to provide k-anonymity and km-anonymity: the original data are transformed by generalizing (i.e., replacing one value with a more abstract one) or suppressing values to achieve the statistical properties required by the anonymization guarantees. Amnesia employs visualization tools and supportive mechanisms to allow non expert users to anonymize relational and object-relational data.

For instructions on depositing or loading data from Dataverse installations to Amnesia, visit https://amnesia.openaire.eu/about-documentation.html

SampleDB

SampleDB is a web-based electronic lab notebook (ELN) with a focus on flexible metadata. SampleDB can export this flexible, process-specific metadata to a new Dataset in a Dataverse installation using the EngMeta Process Metadata block.

For instructions on using the Dataverse export, you can visit https://scientific-it-systems.iffgit.fz-juelich.de/SampleDB/administrator_guide/dataverse_export.html

RedCap

RedCap is a web-based application to capture data for clinical research and create databases and projects.

The *Integrations Dashboard* enables a pull of data from RedCap to a dataset in Dataverse.

GitLab

GitLab is an open source Git repository and platform that provides free open and private repositories, issue-following capabilities, and wikis for collaborative software development.

The *Integrations Dashboard* enables a pull of data from GitLab to a dataset in Dataverse.

iRODS

An open source, metadata driven data management system that is accessible through a host of different clients.

The *Integrations Dashboard* enables a pull of data from iRODS to a dataset in Dataverse.

Integrations Dashboard

The integrations dashboard is software by the Dataverse community to enable easy data transfer from an existing data management platform to a dataset in a Dataverse collection.

Instead of trying to set up Dataverse plug-ins in existing tools and systems to push data to a Dataverse installation, the dashboard works in reverse by being a portal to pull data from tools such as iRODS and GitHub into a dataset.

Its aim is to make integrations more flexible and less dependent on the cooperation of system to integrate with. You can use it to either create a dataset from scratch and add metadata after files have been transferred, or you can use it to compare what is already in an existing dataset to make updating files in datasets easier.

Its goal is to make the dashboard adjustable for a Dataverse installation's needs and easy to connect other systems to as well.

The integrations dashboard is currently in development. A preview and more information can be found at: rdm-integration GitHub repository

Globus

Globus transfer uses an efficient transfer mechanism and has additional features that make it suitable for large files and large numbers of files:

- robust file transfer capable of restarting after network or endpoint failures
- third-party transfer, which enables a user accessing a Dataverse installation in their desktop browser to initiate transfer of their files from a remote endpoint (i.e. on a local high-performance computing cluster), directly to an S3 store managed by the Dataverse installation

Users can transfer files via Globus into and out of datasets, or reference files on a remote Globus endpoint, when their Dataverse installation is configured to use a Globus accessible store(s) and a community-developed dataverse-globus app has been properly installed and configured.

DataLad

DataLad is a free and open source decentralized data management system that is built on git and git-annex and provides a unified interface for version control, deposition, content retrieval, provenance tracking, reproducible execution, and further collaborative management of distributed and arbitrarily large datasets.

If your dataset is structured as a DataLad dataset and you have a local DataLad installation, the datalad-dataverse extension package provides interoperability with Dataverse for the purpose of depositing DataLad datasets to and retrieving DataLad datasets from Dataverse instances, together with full version history.

For further information, visit the `datalad-dataverse` extension's documentation page, see the quickstart for installation details, or follow the step-by-step tutorial to get hands-on experience.

Open OnDemand

Open OnDemand is a web frontend to High Performance Computing (HPC) resources. Through a system called OnDemand Loop, developed at IQSS, researchers can create datasets in Dataverse and upload files to them from their Open OnDemand installation. They can also *download* files from Dataverse.

Galaxy

Import files directly from Dataverse into Galaxy as well as publish datasets containing artifacts (Histories, datasets, etc.) from Galaxy to Dataverse. For details, see https://github.com/galaxyproject/galaxy/pull/19367

3.11.2 Embedding Data on Websites

OpenScholar

OpenScholar is oriented toward hosting websites for academic institutions and offers Dataverse Project Widgets that can be added to web pages. See also:

- *Adding Widgets to an OpenScholar Website* (Dataverse collection level)
- *Adding Widgets to an OpenScholar Website* (dataset level)

3.11.3 Analysis and Computation

Data Explorer

Data Explorer is a GUI which lists the variables in a tabular data file allowing searching, charting and cross tabulation analysis.

For installation instructions, see the *External Tools* section.

Compute Button

The "Compute" button is still highly experimental and has special requirements such as use of a Swift object store, but it is documented under "Setting up Compute" in the *Configuration* section of the Installation Guide.

Binder

Researchers can launch Jupyter Notebooks, RStudio, and other computational environments by entering the DOI of a dataset in a Dataverse installation at https://mybinder.org

A Binder button can also be added to every dataset page to launch Binder from there. Instructions on enabling this feature can be found under *External Tools*.

Additionally, institutions can self host BinderHub (the software that powers mybinder.org), which lists the Dataverse software as one of the supported repository providers.

Renku

Renku is a platform for collaborative data science. It provides a simple, seamless way to bring data, code, and compute together into shared collaborative projects. Users can connect to a variety of "cloud" data sources, including datasets from Dataverse. Datasets are available immediately in interactive compute sessions, without the need to download them first - data is transferred on-demand.

For more information about using published datasets in Renku interactive sessions, have a look at the documentation.

Documentation, tutorials, and how-to guides are available on the Renku Community Portal.

To use Renku, feel free to head to https://renkulab.io - anyone can create an account and there are free public compute resources that you can access.

Avgidea Data Search

Researchers can use a Google Sheets add-on to search for Dataverse installation's CSV data and then import that data into a sheet. See Avgidea Data Search for details.

JupyterHub

The Dataverse-to-JupyterHub Data Transfer Connector streamlines data transfer between Dataverse repositories and the cloud-based platform JupyterHub, enhancing collaborative research. This connector facilitates seamless two-way transfer of datasets and files, emphasizing the potential of an integrated research environment. It is a lightweight client-side web application built using React and relying on the Dataverse External Tool feature, allowing for easy deployment on modern integration systems. Currently, it supports small to medium-sized files, with plans to enable support for large files and signed Dataverse endpoints in the future.

What kind of user is the feature intended for? The feature is intended for researchers, scientists and data analyst who are working with Dataverse instances and JupyterHub looking to ease the data transfer process. See presentation for details.

Rclone

Rclone ("rsync for cloud storage") is a command-line program to sync files and directories to and from different cloud storage providers.

Rclone v1.70+ supports listing and downloading files from Dataverse datasets. For details, consult the Rclone DOI remote documentation and the discussion on the Dataverse mailing list.

Open OnDemand

Open OnDemand is a web frontend to High Performance Computing (HPC) resources. Through a system called OnDemand Loop, developed at IQSS, researchers can click a button to explore a dataset (if the *external tool* has been enabled) or discover and download datasets from Dataverse installations from around the world (and other data repositories) in their Open OnDemand computational environment. From that environment they can also *create datasets* in Dataverse and upload files.

3.11.4 Discoverability

A number of builtin features related to data discovery are listed under *Discoverability* but you can further increase the discoverability of your data by setting up integrations.

Geodisy

Geodisy will take your Dataverse installation's data, search for geospatial metadata and files, and copy them to a new system that allows for visual searching. Your original data and search methods are untouched; you have the benefit of both. For more information, please refer to Geodisy's GitHub Repository.

DataONE

DataONE is a community driven program providing access to data across multiple member repositories, supporting enhanced search and discovery of Earth and environmental data. Membership is free and is most easily achieved by providing schema.org data via science-on-schema.org metadata markup on dataset landing pages, support for which is native in Dataverse. Dataverse installations are welcome join the network to have their datasets included.

3.11.5 Research Data Preservation

Archivematica

Archivematica is an integrated suite of open-source tools for processing digital objects for long-term preservation, developed and maintained by Artefactual Systems Inc. Its configurable workflow is designed to produce system-independent, standards-based Archival Information Packages (AIPs) suitable for long-term storage and management.

Sponsored by the Ontario Council of University Libraries (OCUL), this technical integration enables users of Archivematica to select datasets from connected Dataverse installations and process them for long-term access and digital preservation. For more information and list of known issues, please refer to Artefactual's release notes, integration documentation, and the project wiki.

RDA BagIt (BagPack) Archiving

A Dataverse installation can be configured to submit a copy of published Dataset versions, packaged as Research Data Alliance conformant zipped BagIt bags to Chronopolis via DuraCloud, a local file system, any S3 store, or to Google Cloud Storage. Submission can be automated to occur upon publication, or can be done periodically (via external scripting). The archival status of each Dataset version can be seen in the Dataset page version table and queried via API.

The archival Bags include all of the files and metadata in a given dataset version and are sufficient to recreate the dataset, e.g. in a new Dataverse instance, or potentially in another RDA-conformant repository. Specifically, the archival Bags include an OAI-ORE Map serialized as JSON-LD that describe the dataset and it's files, as well as information about the version of Dataverse used to export the archival Bag.

The DVUploader includes functionality to recreate a Dataset from an archival Bag produced by Dataverse (using the Dataverse API to do so).

For details on how to configure this integration, see *BagIt Export* in the *Configuration* section of the Installation Guide.

3.11.6 Future Integrations

The Dataverse Project Roadmap is a good place to see integrations that the core Dataverse Project team is working on.

If you have an idea for an integration, please ask on the dataverse-community mailing list if someone is already working on it.

Many integrations take the form of "external tools". See the *External Tools* section for details. External tool makers should check out the *Building External Tools* section of the API Guide.

Please help us keep this page up to date making a pull request! To get started, see the *Writing Documentation* section of the Contributor Guide.

3.12 User Administration

This section focuses on user administration tools and tasks.

Contents:

- *Manage Users*

    - *List Users via API*
- *Merge User Accounts*
- *Change User Identifier*
- *Delete a User*
- *Deactivate a User*
- *Confirm Email*
- *Deleting an API Token*
- *Letting Users Manage Notifications*

3.12.1 Manage Users

The Manage Users table gives the network administrator a list of all user accounts in table form. You can access it by clicking the "Manage Users" button on the *Dashboard*, which is linked from the header of all Dataverse installation pages (if you're logged in as an administrator). It lists username, full name, email address, affiliation, the authentication method they use, the roles their account has been granted, and whether or not they have Superuser status.

Users are listed alphabetically by username. The search bar above the table allows you to search for a specific user. It performs a right-truncated wildcard search of the Username, Name, and Email columns. This means, if you search "baseba" then it will search those three columns for any string of text that begins with "baseba", e.g. "baseball" or "baseballfan".

If you would like to assign or remove a user's Superuser status, then you can do so by checking or unchecking their checkbox under the "Superuser" column.

If you would like to remove all roles/permissions from a user's account (in the event of their leaving your organization, for example) then you can do so by clicking the "Remove All" button under the Roles column. This will keep the user's account active, but will revert it to put the account on the level of a default user with default permissions.

List Users via API

There are two ways to list users via API. If you have relatively few users, you can get them all as a dump with this command with a superuser API token:

```
curl -H "X-Dataverse-key: $API_TOKEN" http://localhost:8080/api/admin/authenticatedUsers
```

If you have many users and want to be able to search and paginate through the results, use the command below with a superuser API token:

```
curl -H "X-Dataverse-key: $API_TOKEN" http://localhost:8080/api/admin/list-users
```

With the `list-users` form you can include the following optional query parameters:

- `searchTerm` A string that matches the beginning of a user identifier, first name, last name or email address.
- `itemsPerPage` The number of detailed results to return. The default is 25. This number has no limit. e.g. You could set it to 1000 to return 1,000 results
- `selectedPage` The page of results to return. The default is 1.

3.12.2 Merge User Accounts

See *Merge User Accounts*

3.12.3 Change User Identifier

See *Change User Identifier*

3.12.4 Delete a User

See *Delete a User*

3.12.5 Deactivate a User

See *Deactivate a User*

3.12.6 Confirm Email

A Dataverse installation encourages builtin/local users to verify their email address upon sign up or email change so that sysadmins can be assured that users can be contacted.

The app will send a standard welcome email with a URL the user can click, which, when activated, will store a `lastconfirmed` timestamp in the `authenticateduser` table of the database. Any time this is "null" for a user (immediately after sign up and/or changing of their Dataverse installation email address), their current email on file is considered to not be verified. The link that is sent expires after a time (the default is 24 hours), but this is configurable by a superuser via the `:MinutesUntilConfirmEmailTokenExpires` config option.

Should users' URL token expire, they will see a "Verify Email" button on the account information page to send another URL.

Sysadmins can determine which users have verified their email addresses by looking for the presence of the value `emailLastConfirmed` in the JSON output from listing users (see *Admin* section of Native API in the API Guide). As mentioned in the *Account Creation + Management* section of the User Guide, the email addresses for Shibboleth users are re-confirmed on every login (so their welcome email does not contain a URL to click for this purpose).

3.12.7 Deleting an API Token

If an API token is compromised it should be deleted. Users can generate a new one for themselves as explained in the *Account Creation + Management* section of the User Guide, but you may want to preemptively delete tokens from the database.

Using the API token 7ae33670-be21-491d-a244-008149856437 as an example:

```
delete from apitoken where tokenstring = '7ae33670-be21-491d-a244-008149856437';
```

You should expect the output `DELETE 1` after issuing the command above.

3.12.8 Letting Users Manage Notifications

See *Notifications* in the User Guide for how notifications are described to end users.

You can let users manage which notification types they wish to receive by setting *:ShowMuteOptions* to "true":

```
curl -X PUT -d 'true' http://localhost:8080/api/admin/settings/:ShowMuteOptions
```

This enables additional settings for each user in the notifications tab of their account page. The users can select which in-app notifications and/or e-mails they wish to receive out of the following list:

- `APIGENERATED` API token is generated
- `ASSIGNROLE` Role is assigned
- `CHECKSUMFAIL` Checksum validation failed
- `CHECKSUMIMPORT` Dataset had file checksums added via a batch job
- `CONFIRMEMAIL` Email Verification
- `CREATEACC` Account is created
- `CREATEDS` Your dataset is created
- `CREATEDV` Dataverse collection is created
- `DATASETCREATED` Dataset was created by user
- `DATASETMENTIONED` Dataset was mentioned via *Linked Data Notification API*
- `DATASETMOVED` Dataset was moved by user
- `FILESYSTEMIMPORT` Dataset has been successfully uploaded and verified
- `GRANTFILEACCESS` Access to file is granted
- `INGESTCOMPLETEDWITHERRORS` Ingest completed with errors
- `INGESTCOMPLETED` Ingest is completed
- `PUBLISHEDDS` Dataset is published
- `PUBLISHFAILED_PIDREG` Publish has failed
- `REJECTFILEACCESS` Access to file is rejected
- `REQUESTFILEACCESS` Access to file is requested
- `RETURNEDDS` Returned from review
- `REVOKEROLE` Role is revoked
- `STATUSUPDATED` Status of dataset has been updated
- `SUBMITTEDDS` Submitted for review
- `WORKFLOW_FAILURE` External workflow run has failed
- `WORKFLOW_SUCCESS` External workflow run has succeeded
- `PIDRECONCILED` Dataset persistent identifier changed

After enabling this feature, all notifications are enabled by default, until this is changed by the user.

You can shorten this list by configuring some notification types (e.g., `ASSIGNROLE` and `REVOKEROLE`) to be always muted for everyone and not manageable by users (not visible in the user interface) with the *:AlwaysMuted* setting:

```
curl -X PUT -d 'ASSIGNROLE,REVOKEROLE' http://localhost:8080/api/admin/settings/
:AlwaysMuted
```

Finally, you can set some notifications (e.g., REQUESTFILEACCESS, GRANTFILEACCESS and REJECTFILEACCESS) as always enabled for everyone and not manageable by users (grayed out in the user interface) with the *:NeverMuted* setting:

```
curl -X PUT -d 'REQUESTFILEACCESS,GRANTFILEACCESS,REJECTFILEACCESS' http://
localhost:8080/api/admin/settings/:NeverMuted
```

3.13 Managing Datasets and Dataverse Collections

Contents:

- *Dataverse Collections*
  - *Delete a Dataverse Collection*
  - *Move a Dataverse Collection*
  - *Link a Dataverse Collection*
  - *Unlink a Dataverse Collection*
  - *List Dataverse Collection Links*
  - *Add Dataverse Collection RoleAssignments to Dataverse Subcollections*
  - *Configure a Dataverse Collection to Store All New Files in a Specific File Store*
  - *Configure a Dataverse Collection to Allow Use of a Given Curation Label Set*
  - *Configure a Collection for Review Datasets*
    - *Load the Review Metadata Block*
    - *Create and Enable Custom "Rubric" Metadata Blocks*
    - *Create a Review Dataset Type*
    - *Create a Collection for Reviews and Configure Permissions*
    - *Allow the Review Dataset Type for the Collection*
    - *Invite Users to Create Review Datasets*
- *Datasets*
  - *Move a Dataset*
  - *Link a Dataset*
  - *List Collections that are Linked from a Dataset*
  - *Unlink a Dataset*
  - *Mint a PID for a File That Does Not Have One*
  - *Mint PIDs for all unregistered published files in the specified collection*
  - *Mint PIDs for ALL unregistered files in the database*
  - *Mint a New DOI for a Dataset with a Handle*
  - *Update Target URL for a Published Dataset at the PID provider*

- *Update Target URL for all Published Datasets at the PID provider*
- *Update Metadata for a Published Dataset at the PID provider*
- *Update Metadata for all Published Datasets at the PID provider*
- *Check for Unreserved PIDs and Reserve Them*
- *Make Metadata Updates Without Changing Dataset Version*
- *Diagnose Constraint Violations Issues in Datasets*
- *Configure a Dataset to Store All New Files in a Specific File Store*
- *Configure a Dataset to Allow Use of a Curation Label Set*

3.13.1 Dataverse Collections

Delete a Dataverse Collection

Dataverse collections have to be empty to delete them. Navigate to the Dataverse collection and click "Edit" and then "Delete Dataverse" to delete it. To delete a Dataverse collection via API, see the *Native API* section of the API Guide.

Move a Dataverse Collection

Moves a Dataverse collection whose id is passed to an existing Dataverse collection whose id is passed. The Dataverse collection alias also may be used instead of the id. If the moved Dataverse collection has a guestbook, template, metadata block, link, or featured Dataverse collection that is not compatible with the destination Dataverse collection, you will be informed and given the option to force the move and remove the association. Only accessible to superusers.

```
curl -H "X-Dataverse-key: $API_TOKEN" -X POST http://$SERVER/api/dataverses/$id/move/
↪$destination-id
```

Link a Dataverse Collection

Creates a link between a Dataverse collection and another Dataverse collection (see the *Dataverse Collection Linking* section of the User Guide for more information). Only accessible to superusers.

```
curl -H "X-Dataverse-key: $API_TOKEN" -X PUT http://$SERVER/api/dataverses/$linked-
↪dataverse-alias/link/$linking-dataverse-alias
```

Unlink a Dataverse Collection

Removes a link between a Dataverse collection and another Dataverse collection. Only accessible to superusers.

```
curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE http://$SERVER/api/dataverses/$linked-
↪dataverse-alias/deleteLink/$linking-dataverse-alias
```

List Dataverse Collection Links

Provides information about whether a certain Dataverse collection ($dataverse-alias) is linked to or links to another collection. Only accessible to superusers.

```
curl -H "X-Dataverse-key:$API_TOKEN" http://$SERVER/api/dataverses/$dataverse-alias/links
```

Add Dataverse Collection RoleAssignments to Dataverse Subcollections

Recursively assigns the users and groups having a role(s),that are in the set configured to be inheritable via the :InheritParentRoleAssignments setting, on a specified Dataverse collections to have the same role assignments on all of the Dataverse collections that have been created within it. The response indicates success or failure and lists the individuals/groups and Dataverse collections involved in the update. Only accessible to superusers.

```
curl -H "X-Dataverse-key: $API_TOKEN" http://$SERVER/api/admin/dataverse/$dataverse-
↪alias/addRoleAssignmentsToChildren
```

Configure a Dataverse Collection to Store All New Files in a Specific File Store

To direct new files (uploaded when datasets are created or edited) for all datasets in a given Dataverse collection, the store can be specified via the API as shown below, or by editing the 'General Information' for a Dataverse collection on the Dataverse collection page. Requires permission to edit the Dataverse collection (for example, the `EditDataverse` permission).

```
curl -H "X-Dataverse-key: $API_TOKEN" -X PUT -d $storageDriverLabel http://$SERVER/api/
↪dataverses/$dataverse-alias/storageDriver
```

(Note that for `dataverse.files.store1.label=MyLabel`, you should pass `MyLabel`.)

A store assigned directly to a collection can be seen using:

```
curl -H "X-Dataverse-key: $API_TOKEN" http://$SERVER/api/dataverses/$dataverse-alias/
↪storageDriver
```

This may be null. To get the effective storageDriver for a collection, which may be inherited from a parent collection or be the installation default, you can use:

```
curl -H "X-Dataverse-key: $API_TOKEN" http://$SERVER/api/dataverses/$dataverse-alias/
↪storageDriver?getEffective=true
```

This will never be null.

(Note that for `dataverse.files.store1.label=MyLabel`, the JSON response will include "name":"store1" and "label":"MyLabel".)

To delete a store assigned directly to a collection (so that the colllection's effective store is inherted from it's parent or is the global default), use:

```
curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE http://$SERVER/api/dataverses/$dataverse-
↪alias/storageDriver
```

The available drivers within a collection can be listed with:

```
curl -H "X-Dataverse-key: $API_TOKEN" http://$SERVER/api/dataverses/$dataverse-alias/
↪allowedStorageDrivers
```

(Individual datasets can be configured to use specific file stores as well. See the "Datasets" section below.)

Configure a Dataverse Collection to Allow Use of a Given Curation Label Set

Datasets within a given Dataverse collection can be annotated with a Curation Label to indicate the status of the dataset with respect to a defined curation process. Labels are completely customizable (alphanumeric or spaces, up to 32 characters, e.g. "Author contacted", "Privacy Review", "Awaiting paper publication").

The label is applied to a draft Dataset version via the user interface or API and the available label sets are defined by *:AllowedCurationLabels*. Internally, the labels have no effect, and at publication, any existing label will be removed. A reporting API call allows admins to get a list of datasets and their curation statuses.

The label set used for a collection can be specified via the API as shown below, or by editing the 'General Information' for a Dataverse collection on the Dataverse collection page. Only accessible to superusers.

The curationLabelSet to use within a given collection can be set by specifying its name using:

```
curl -H "X-Dataverse-key: $API_TOKEN" -X PUT http://$SERVER/api/admin/dataverse/
↪$dataverse-alias/curationLabelSet?name=$curationLabelSetName
```

The reserved word "DISABLED" can be used to disable this feature within a given Dataverse collection.

The name of the current curationLabelSet can be seen using:

```
curl -H "X-Dataverse-key: $API_TOKEN" http://$SERVER/api/admin/dataverse/$dataverse-
↪alias/curationLabelSet
```

and can be reset to the default (inherited from the parent collection or DISABLED for the root collection) with:

```
curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE http://$SERVER/api/admin/dataverse/
↪$dataverse-alias/curationLabelSet
```

The available curation label sets can be listed with:

```
curl -H "X-Dataverse-key: $API_TOKEN" http://$SERVER/api/admin/dataverse/
↪curationLabelSets
```

If the :AllowedCurationLabels setting has a value, one of the available choices will always be "DISABLED" which allows curation labels to be turned off for a given collection/dataset.

Individual datasets can be configured to use specific curationLabelSets as well. See the "Datasets" section below.

Configure a Collection for Review Datasets

Review Datasets are a specialized type of dataset that can be used to review resources (such as datasets) in the Dataverse installation itself or resources in external data repositories.

Review datasets require some setup, as described below.

Load the Review Metadata Block

First, download the Review metadata block tsv file from *Experimental Metadata*.

Then, load the block and update Solr. See the following sections of *Metadata Customization* for details:

- *Loading TSV files into a Dataverse Installation*
- *Updating the Solr Schema*

Create and Enable Custom "Rubric" Metadata Blocks

The Review metadata block gives you a few basic fields common to all reviews such as the URL of the item being reviewed.

You probably will want to create your own metadata blocks specific to the resources you are reviewing, your own "rubric". See *Metadata Customization* for details on creating and enabling custom metadata blocks.

Instead of creating a new custom metadata block from scratch (if you simply want to evaluate the feature, for example), in a test environment, you can use the "Trusted Data Dimensions and Intensities" for testing. (A test environment is advised because metadata blocks cannot be deleted once they are loaded (https://github.com/IQSS/dataverse/issues/9628).) These are the files to download:

- `rubric_trusteddatadimensionsintensities.tsv`
- `rubric_trusteddatadimensionsintensities.properties` (optional)

After loading the block, don't forget to update the Solr schema!

As in the example above, the metadata block must start with `rubric_` (the "metadataBlock name" in the tsv itself) to be included in the output of the *List Reviews* API endpoint.

Create a Review Dataset Type

Review datasets are built on the *Dataset Types* feature. Dataset types can only be created via API so follow the steps under *Add Dataset Type*. Copy and paste from below or download `review.json` and pass it to the API.

```
{
  "name": "review",
  "displayName": "Review",
  "description": "A review of a dataset compiled by the expert community.",
  "linkedMetadataBlocks": [
    "review"
  ]
}
```

Create a Collection for Reviews and Configure Permissions

Follow the normal steps:

- *Create a New Dataverse Collection*.
- *Roles & Permissions*.

Allow the Review Dataset Type for the Collection

Non-dataset types, such as the "review" type, are only available when a collection admin has enabled them, via API.

Using the API *Change Collection Attributes*, change the `allowedDatasetTypes` attribute so that it includes "review". If you only want to allow reviews, you can pass just `review`. If you want to allow multiple dataset types, you can pass a comma-separated list, such as `review,dataset`.

Invite Users to Create Review Datasets

At this point, users should be able to create review datasets via API, if you gave them permission on the collection. You can point them to *Creating a Review Dataset* for details.

3.13.2 Datasets

Move a Dataset

Superusers can move datasets using the dashboard. See also *Dashboard*.

Moves a dataset whose id is passed to a Dataverse collection whose alias is passed. If the moved dataset has a guestbook or a Dataverse collection link that is not compatible with the destination Dataverse collection, you will be informed and given the option to force the move (with `forceMove=true` as a query parameter) and remove the guestbook or link (or both). Only accessible to users with permission to publish the dataset in the original and destination Dataverse collection. Note: any roles granted to users on the dataset will continue to be in effect after the dataset has been moved.

```
curl -H "X-Dataverse-key: $API_TOKEN" -X POST http://$SERVER/api/datasets/$id/move/$alias
```

Link a Dataset

Creates a link between a dataset and a Dataverse collection (see the *Dataset Linking* section of the User Guide for more information). Accessible to users with Link Dataset permission on the Dataverse collection.

```
curl -H "X-Dataverse-key: $API_TOKEN" -X PUT http://$SERVER/api/datasets/$linked-dataset-
↪id/link/$linking-dataverse-alias
```

List Collections that are Linked from a Dataset

Lists the link(s) created between a dataset and a Dataverse collection (see the *Dataset Linking* section of the User Guide for more information).

```
curl -H "X-Dataverse-key: $API_TOKEN" http://$SERVER/api/datasets/$linked-dataset-id/
↪links
```

It returns a list in the following format (new format as of v6.4):

```
{
  "status": "OK",
  "data": {
    "id": 5,
    "identifier": "FK2/OTCWMM",
    "linked-dataverses": [
      {
        "id": 2,
        "alias": "dataverse1",
        "displayName": "Lab experiments 2023 June"
      }
    ]
  }
}
```

Unlink a Dataset

Removes a link between a dataset and a Dataverse collection. Accessible to users with Link Dataset permission on the Dataverse collection.

```
curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE http://$SERVER/api/datasets/$linked-
↪dataset-id/deleteLink/$linking-dataverse-alias
```

Mint a PID for a File That Does Not Have One

In the following example, the database id of the file is 42:

```
export FILE_ID=42
curl "http://localhost:8080/api/admin/$FILE_ID/registerDataFile"
```

This method will return a FORBIDDEN response if minting of file PIDs is not enabled for the collection the file is in. (Note that it is possible to have file PIDs enabled for a specific collection, even when it is disabled for the Dataverse installation as a whole. See *Change Collection Attributes* in the Native API Guide.)

Mint PIDs for all unregistered published files in the specified collection

The following API will register the PIDs for all the yet unregistered published files in the datasets **directly within the collection** specified by its alias:

```
curl "http://localhost:8080/api/admin/registerDataFiles/{collection_alias}"
```

It will not attempt to register the datafiles in its sub-collections, so this call will need to be repeated on any sub-collections where files need to be registered as well. File-level PID registration must be enabled on the collection. (Note that it is possible to have it enabled for a specific collection, even when it is disabled for the Dataverse installation as a whole. See *Change Collection Attributes* in the Native API Guide.)

This API will sleep for 1 second between registration calls by default. A longer sleep interval can be specified with an optional `sleep=` parameter:

```
curl "http://localhost:8080/api/admin/registerDataFiles/{collection_alias}?sleep=5"
```

Mint PIDs for ALL unregistered files in the database

The following API will attempt to register the PIDs for all the published files in your instance, in collections that allow file PIDs, that do not yet have them:

```
curl http://localhost:8080/api/admin/registerDataFileAll
```

The application will attempt to sleep for 1 second between registration attempts as not to overload your persistent identifier service provider. Note that if you have a large number of files that need to be registered in your Dataverse, you may want to consider minting file PIDs within indivdual collections, or even for individual files using the `registerDataFiles` and/or `registerDataFile` endpoints above in a loop, with a longer sleep interval between calls.

Mint a New DOI for a Dataset with a Handle

Mints a new identifier for a dataset previously registered with a handle. Only accessible to superusers.

```
curl -H "X-Dataverse-key: $API_TOKEN" -X POST http://$SERVER/api/admin/$dataset-id/
↪reregisterHDLToPID
```

Update Target URL for a Published Dataset at the PID provider

Forces update to the target URL provided to the PID provider of a published dataset and assures the PID is findable. Only accessible to superusers.

```
curl -H "X-Dataverse-key: $API_TOKEN" -X POST http://$SERVER/api/datasets/$dataset-id/
↪modifyRegistration
```

Update Target URL for all Published Datasets at the PID provider

Forces update to the target URL provided to the PID provider of all published datasets and assures the PID is findable. Only accessible to superusers.

```
curl -H "X-Dataverse-key: $API_TOKEN" -X POST http://$SERVER/api/datasets/
↪modifyRegistrationAll
```

Update Metadata for a Published Dataset at the PID provider

Checks to see that the PID metadata for a published dataset (and any released files in it using file PIDs) is up-to-date at the provider and updates the metadata if necessary. Only accessible to superusers.

```
curl -H "X-Dataverse-key: $API_TOKEN" -X POST http://$SERVER/api/datasets/$dataset-id/
↪modifyRegistrationMetadata
```

Update Metadata for all Published Datasets at the PID provider

Checks to see that the PID metadata is up-to-date at the provider for all published datasets (and any released files in them using file PIDs) and updates the metadata if necessary. Only accessible to superusers.

```
curl -H "X-Dataverse-key: $API_TOKEN" -X POST http://$SERVER/api/datasets/
↪modifyRegistrationPIDMetadataAll
```

The call returns 200/OK as long as the call completes. Any errors for individual datasets are reported in the log.

Check for Unreserved PIDs and Reserve Them

See *PIDs* in the API Guide for details.

Make Metadata Updates Without Changing Dataset Version

As a superuser, click "Update Current Version" when publishing. (This option is only available when a 'Minor' update would be allowed.)

Diagnose Constraint Violations Issues in Datasets

To identify invalid data values in specific datasets (if, for example, an attempt to edit a dataset results in a ConstraintViolationException in the server log), or to check all the datasets in the Dataverse installation for constraint violations, see *Dataset Validation* in the *Native API* section of the User Guide.

Configure a Dataset to Store All New Files in a Specific File Store

Configure an individual dataset to use a specific file store (this API can only be used by a superuser)

```
curl -H "X-Dataverse-key: $API_TOKEN" -X PUT -d $storageDriverLabel http://$SERVER/api/
↪datasets/$dataset-id/storageDriver
```

The effective store can be seen using:

```
curl http://$SERVER/api/datasets/$dataset-id/storageDriver
```

The output of the API will include the id, label, type (for example, "file" or "s3") as well as the support for direct download and upload.

To remove an assigned store, and allow the dataset to inherit the store from it's parent collection, use the following (only a superuser can do this)

```
curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE http://$SERVER/api/datasets/$dataset-id/
↪storageDriver
```

The available drivers can be listed with:

```
curl -H "X-Dataverse-key: $API_TOKEN" http://$SERVER/api/admin/dataverse/storageDrivers
```

Configure a Dataset to Allow Use of a Curation Label Set

A dataset can be annotated with a Curation Label to indicate the status of the dataset with respect to a defined curation process. Labels are completely customizable (alphanumeric or spaces, up to 32 characters, e.g. "Author contacted", "Privacy Review", "Awaiting paper publication").

The label is applied to a draft Dataset version via the user interface or API and the available label sets are defined by *:AllowedCurationLabels*. Internally, the labels have no effect, and at publication, any existing label will be removed. A reporting API call allows admins to get a list of datasets and their curation statuses.

The label set used for a dataset can be specified via the API as shown below. Only accessible to superusers.

The curationLabelSet to use within a given dataset can be set by specifying its name using:

```
curl -H "X-Dataverse-key: $API_TOKEN" -X PUT http://$SERVER/api/datasets/$dataset-id/
↪curationLabelSet?name=$curationLabelSetName
```

The reserved word "DISABLED" can be used to disable this feature within a given Dataverse collection.

The name of the current curationLabelSet can be seen using:

```
curl -H "X-Dataverse-key: $API_TOKEN" http://$SERVER/api/datasets/$dataset-id/
↪curationLabelSet
```

and can be reset to the default (inherited from the parent collection) with (only a superuser can do this)

```
curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE http://$SERVER/api/datasets/$dataset-id/
↪curationLabelSet
```

The available curationLabelSets can be listed with:

```
curl -H "X-Dataverse-key: $API_TOKEN" http://$SERVER/api/admin/dataverse/
↪curationLabelSets
```

If the :AllowedCurationLabels setting has a value, one of the available choices will always be "DISABLED" which allows curation labels to be turned off for a given collection/dataset.

Collections can be configured to use specific curationLabelSets as well. See the "Dataverse Collections" section above.

3.14 Solr Search Index

A Dataverse installation requires Solr to be operational at all times. If you stop Solr, you should see an error about this on the root Dataverse installation page, which is powered by the search index Solr provides. You can set up Solr by following the steps in our Installation Guide's *Prerequisites* and *Configuration* sections explaining how to configure it. This section you're reading now is about the care and feeding of the search index. PostgreSQL is the "source of truth" and the Dataverse installation will copy data from PostgreSQL into Solr. For this reason, the search index can be rebuilt at any time. Depending on the amount of data you have, this can be a slow process. You are encouraged to experiment with production data to get a sense of how long a full reindexing will take.

Contents:

- *Full Reindex*
  - *Clear and Reindex*
    - *Index and Database Consistency*
    - *Clearing ALL Data from Solr*
    - *Start Async Reindex*
  - *Reindex in Place*
    - *Clear Index Timestamps*
    - *Start or Continue Async Reindex*
- *Manual Reindexing*
  - *Reindexing Dataverse Collections*
  - *Reindexing Datasets*
    - *Clearing a Dataset from Solr*
- *Reindexing Permissions*
  - *Reindexing Permissions for All Objects*
  - *Reindexing Permissions for a Single Object*
- *Manually Querying Solr*

3.14.1 Full Reindex

There are two ways to perform a full reindex of the Dataverse installation search index. Starting with a "clear" ensures a completely clean index but involves downtime. Reindexing in place doesn't involve downtime but does not ensure a completely clean index (e.g. stale entries from destroyed datasets can remain in the index).

Clear and Reindex

Index and Database Consistency

Get a list of all database objects that are missing in Solr, and Solr documents that are missing in the database:

```
curl http://localhost:8080/api/admin/index/status
```

Remove all Solr documents that are orphaned (i.e. not associated with objects in the database):

```
curl http://localhost:8080/api/admin/index/clear-orphans
```

Clearing ALL Data from Solr

Please note that the moment you issue this command, it will appear to end users looking at the root Dataverse installation page that all data is gone! This is because the root Dataverse installation page is powered by the search index.

```
curl http://localhost:8080/api/admin/index/clear
```

Start Async Reindex

Please note that this operation may take hours depending on the amount of data in your system and whether or not you installation is using full-text indexing. More information on this, as well as some reference times, can be found at https://github.com/IQSS/dataverse/issues/50.

```
curl http://localhost:8080/api/admin/index
```

Reindex in Place

An alternative to completely clearing the search index is to reindex in place.

Clear Index Timestamps

```
curl -X DELETE http://localhost:8080/api/admin/index/timestamps
```

Start or Continue Async Reindex

If indexing stops, this command should pick up where it left off based on which index timestamps have been set, which is why we start by clearing these timestamps above. These timestamps are stored in the `dvobject` database table.

```
curl http://localhost:8080/api/admin/index/continue
```

3.14.2 Manual Reindexing

If you have made manual changes to a dataset in the database or wish to reindex a dataset that Solr didn't want to index properly, it is possible to manually reindex Dataverse collections and datasets.

Reindexing Dataverse Collections

Dataverse collections must be referenced by database object ID. If you have direct database access an SQL query such as

```
select id from dataverse where alias='dataversealias';
```

should work, or you may click the Dataverse Software's "Edit" menu and look for *dataverseId*= in the URLs produced by the drop-down. Then, to re-index:

```
curl http://localhost:8080/api/admin/index/dataverses/135
```

which should return: *{"status":"OK","data":{"message":"starting reindex of dataverse 135"}}*

Reindexing Datasets

Datasets may be referenced by persistent ID or by database object ID. To re-index by persistent ID:

```
curl http://localhost:8080/api/admin/index/dataset?persistentId=doi:10.5072/FK2/AAA000
```

To re-index a dataset by its database ID:

```
curl http://localhost:8080/api/admin/index/datasets/7504557
```

Clearing a Dataset from Solr

This API will clear the Solr entry for the dataset specified. It can be useful if you have reasons to want to hide a published dataset from showing in search results and/or on Collection pages, but don't want to destroy and purge it from the database just yet.

```
curl -X DELETE http://localhost:8080/api/admin/index/datasets/<DATABASE_ID>
```

This can be reversed of course by re-indexing the dataset with the API above.

3.14.3 Reindexing Permissions

It is possible to just re-index the permissions on Solr entries without re-indexing the content.

Reindexing Permissions for All Objects

Re-index permissions for all Dataverse collections, datasets, and files. This is an asynchronous operation that may take a long time to complete on large installations. This endpoint requires the API token of a superuser.

```
export SERVER_URL=https://demo.dataverse.org
curl -X POST -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/admin/index/perms"
```

If indexing is already in progress, the API will return a 409 ("CONFLICT") response. Otherwise, it will return a 200 ("OK") response and the indexing process will start in the background. Check the server logs for progress.

Reindexing Permissions for a Single Object

It is also possible to re-index permissions for a single object by database ID:

```
export SERVER_URL=https://demo.dataverse.org
export ID=42
curl -X POST -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/admin/index/perms/$ID"
```

This operation is performed synchronously.

3.14.4 Manually Querying Solr

If you suspect something isn't indexed properly in Solr, you may bypass the Dataverse installation's web interface and query the command line directly to verify what Solr returns:

```
curl "http://localhost:8983/solr/collection1/select?q=dsPersistentId:doi:10.15139/S3/
HFV0AO"
```

to see the JSON you were hopefully expecting to see passed along to the Dataverse installation.

3.15 IP Groups

IP Groups can be used to permit download of restricted files by IP addresses rather than people. For example, you may want to allow restricted files to be downloaded by researchers who physically enter a library and make use of the library's network.

Contents:

- *Listing IP Groups*
- *Creating an IP Group*
- *Listing an IP Group*
- *Deleting an IP Group*

3.15.1 Listing IP Groups

IP Groups can be listed with the following curl command:

```
curl http://localhost:8080/api/admin/groups/ip
```

3.15.2 Creating an IP Group

IP Groups must be expressed as ranges in IPv4 or IPv6 format. For illustrative purposes, here is a example of the entire IPv4 and IPv6 range that you can `download` and edit to have a narrower range to meet your needs. If you need your IP Group to only encompass a single IP address, you must enter that IP address for the "start" and "end" of the range. If you don't use IPv6 addresses, you can delete that section of the JSON. Please note that the "alias" must be unique if you define multiple IP Groups. You should give it a meaningful "name" since both "alias" and "name" will appear and be searchable in the GUI when your users are assigning roles.

```
{
  "alias": "ipGroupAll",
  "name": "IP group to match all IPv4 and IPv6 addresses",
  "ranges": [
    [
      "0.0.0.0",
      "255.255.255.255"
    ],
    [
      "::",
      "ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff"
    ]
  ]
}
```

Let's say you download the example above and edit it to give it a range used by your library, giving it a filename of `ipGroup1.json` and putting it in the `/tmp` directory. Next, load it into your Dataverse installation using the following curl command:

```
curl -X POST -H 'Content-type:  application/json' http://localhost:8080/api/admin/groups/
ip --upload-file /tmp/ipGroup1.json
```

Note that you can update a group the same way, as long as you use the same alias.

3.15.3 Listing an IP Group

Let's say you used "ipGroup1" as the alias of the IP Group you created above. To list just that IP Group, you can include the alias in the curl command like this:

```
curl http://localhost:8080/api/admin/groups/ip/ipGroup1
```

3.15.4 Deleting an IP Group

It is not recommended to delete an IP Group that has been assigned roles. If you want to delete an IP Group, you should first remove its permissions.

To delete an IP Group with an alias of "ipGroup1", use the curl command below:

```
curl -X DELETE http://localhost:8080/api/admin/groups/ip/ipGroup1
```

3.16 Mail Domain Groups

Groups can be defined based on the domain part of users (verified) email addresses. Email addresses that match one or more groups configuration will add the user to them.

Within the scientific community, in many cases users will use a institutional email address for their account in a Dataverse installation. This might offer a simple solution for building groups of people, as the domain part can be seen as a selector for group membership.

Some use cases: installations that like to avoid Shibboleth, enable self sign up, offer multi-tenancy or can't use *IP Groups* plus many more.

Hint

Please be aware that non-verified mail addresses will exclude the user even if matching. This is to avoid privilege escalation.

Contents:

- *Listing Mail Domain Groups*
- *Listing a specific Mail Domain Group*
- *Creating a Mail Domain Group*
  - *Matching with Domains or Regular Expressions*
- *Updating a Mail Domain Group*
- *Deleting a Mail Domain Group*

3.16.1 Listing Mail Domain Groups

Mail Domain Groups can be listed with the following curl command:

```
curl http://localhost:8080/api/admin/groups/domain
```

3.16.2 Listing a specific Mail Domain Group

Let's say you used "domainGroup1" as the alias of the Mail Domain Group you created below. To list just that Mail Domain Group, you can include the alias in the curl command like this:

```
curl http://localhost:8080/api/admin/groups/domain/domainGroup1
```

3.16.3 Creating a Mail Domain Group

Mail Domain Groups can be created with a simple JSON file such as domainGroup1.json:

```
{
  "name": "Users from @example.org",
  "alias": "exampleorg",
  "description": "Any verified user from Example Org will be included in this group.",
  "domains": ["example.org"]
}
```

Giving a `description` is optional. The `name` will be visible in the permission UI, so be sure to pick a sensible value.

The `domains` field is mandatory to be an array. This enables creation of multi-domain groups, too.

Obviously you can create as many of these groups you might like, as long as the `alias` is unique.

To load it into your Dataverse installation, either use a `POST` or `PUT` request (see below):

```
curl -X POST -H 'Content-type: application/json' http://localhost:8080/api/admin/groups/
domain --upload-file domainGroup1.json
```

Matching with Domains or Regular Expressions

Adding simple domain names requires exact matches of user email domains and the configured domains of a group. Although you could add multiple domains to a group, those still require exact matches.

You can also use one or multiple regular expressions instead of simple domains for a group. Those should not be mixed, although it would work. Regular expressions still require exact matches, but are much more flexible and are designed to support installation-specific use cases for group management.

Some hints:

- Due to their excessive CPU usage, regular expressions should be used rarely.
- Remember to properly escape "\" in your regular expression. Both Java and JSON are a bit picky about this. E.g. a character class "\d" would have to be escaped as "\\d". Plenty of tutorials on the web explain this in more detail.
- There is no way the Dataverse Software can detect a wrong regular expression for you. Be sure to do extensive testing, as a misconfigured group could result in privilege escalation or an unexpected influx of support contacts.
- Remember to enable the regular expression support for a group by adding `"regex": true`!

A short example for a group using regular expressions:

```
{
  "name": "Users from @example.org",
  "alias": "exampleorg-regex",
  "description": "Any verified user from x@example.org or x@sub.example.org will be␣
↪included.",
  "regex": true,
  "domains": ["example\\.org", "[a-z]+\\.example\\.org"]
}
```

3.16.4 Updating a Mail Domain Group

Editing a group is done by replacing it. Grab your group definition like the domainGroup1.json example above, change it as you like and PUT it into your installation:

```
curl -X PUT -H 'Content-type:  application/json' http://localhost:8080/api/admin/groups/
domain/domainGroup1 --upload-file domainGroup1.json
```

Please make sure that the alias of the group you want to change is included in the path. You also need to ensure that this alias matches with the one given in your JSON file.

> **Hint**
>
> This is an idempotent call, so it will create the group given if not present.

3.16.5 Deleting a Mail Domain Group

To delete a Mail Domain Group with an alias of "domainGroup1", use the curl command below:

```
curl -X DELETE http://localhost:8080/api/admin/groups/domain/domainGroup1
```

Please note: it is not recommended to delete a Mail Domain Group that has been assigned roles. If you want to delete a Mail Domain Group, you should first remove its permissions.

3.17 Storage Quotas for Collections

Please note that this is a new and still experimental feature (as of Dataverse v6.1 release).

Instance admins can now define storage quota limits for specific collections. These limits can be set, changed and/or deleted via the provided APIs (please see the *Update Collection Input Levels* section of the *Native API* guide). The Read version of the API is available to the individual collection admins (i.e., a collection owner can check on the quota configured for their collection), but only superusers can set, change or disable storage quotas.

Storage quotas are *inherited* by subcollections. In other words, when storage use limit is set for a specific collection, it applies to all the datasets immediately under it and in its sub-collections, unless different quotas are defined there and so on. Each file added to any dataset in that hierarchy counts for the purposes of the quota limit defined for the top collection. A storage quota defined on a child sub-collection overrides whatever quota that may be defined on the parent, or inherited from an ancestor.

For example, a collection `A` has the storage quota set to 10GB. It has 3 sub-collections, `B`, `C` and `D`. Users can keep uploading files into the datasets anywhere in this hierarchy until the combined size of 10GB is reached between them. However, if an admin has reasons to limit one of the sub-collections, `B` to 3GB only, that quota can be explicitly set there. This both limits the growth of `B` to 3GB, and also *guarantees* that allocation to it. I.e. the contributors to collection `B` will be able to keep adding data until the 3GB limit is reached, even after the parent collection `A` reaches the combined 10GB limit (at which point `A` and all its subcollections except for `B` will become read-only).

We do not yet know whether this is going to be a popular, or needed use case - a child collection quota that is different from the quota it inherits from a parent. It is likely that for many instances it will be sufficient to be able to define quotas for collections and have them apply to all the child objects underneath. We will examine the response to this feature and consider making adjustments to this scheme based on it. We are already considering introducing other types of quotas, such as limits by users or specific storage volumes.

Please note that only the sizes of the main datafiles and the archival tab-delimited format versions, as produced by the ingest process are counted for the purposes of enforcing the limits. Automatically generated "auxiliary" files, such as rescaled image thumbnails and metadata exports for datasets are not.

When quotas are set and enforced, the users will be informed of the remaining storage allocation on the file upload page together with other upload and processing limits.

Part of the new and experimental nature of this feature is that we don't know for the fact yet how well it will function in real life on a very busy production system, despite our best efforts to test it prior to the release. One specific issue is having to update the recorded storage use for every parent collection of the given dataset whenever new files are added. This includes updating the combined size of the root, top collection - which will need to be updated after *every* file upload. In an unlikely case that this will start causing problems with race conditions and database update conflicts, it is possible to disable these updates (and thus disable the storage quotas feature), by setting the *dataverse.storageuse.disable-storageuse-increments* JVM setting to true.

3.18 Monitoring

Once you're in production, you'll want to set up some monitoring. This page may serve as a starting point for you but you are encouraged to share your ideas with the Dataverse community! You may also be interested in the *Performance* section of the Developer Guide.

Contents:

- *Operating System Monitoring*
  - *Munin*
- *HTTP Traffic*
  - *Monitoring HTTP Traffic from the Client Side*
  - *Monitoring HTTP Traffic from the Server Side*
  - *AWStats*
- *Database Connection Pool Used by App Server*
- *actionlogrecord*
  - *An Important Note about ActionLogRecord Table:*
- *Edit Draft Versions Logging*
- *Solr Indexing Failures Logging*
- *EJB Timers*
- *AWS RDS*
- *MicroProfile Metrics endpoint*

3.18.1 Operating System Monitoring

In production you'll want to monitor the usual suspects such as CPU, memory, free disk space, etc. There are a variety of tools in this space but we'll highlight Munin below because it's relatively easy to set up.

Munin

https://munin-monitoring.org says, "A default installation provides a lot of graphs with almost no work." From RHEL or CentOS 7, you can try the following steps.

Enable the EPEL yum repo (if you haven't already):

```
yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

Install Munin:

```
yum install munin
```

Start Munin:

```
systemctl start munin-node.service
```

Configure Munin to start at boot:

```
systemctl enable munin-node.service
```

Create a username/password (i.e. "admin" for both):

```
htpasswd /etc/munin/munin-htpasswd admin
```

Assuming you are fronting your app server with Apache, prevent Apache from proxying "/munin" traffic to the app server by adding the following line to your Apache config:

```
ProxyPassMatch ^/munin !
```

Then reload Apache to pick up the config change:

```
systemctl reload httpd.service
```

Test auth for the web interface:

```
curl http://localhost/munin/ -u admin:admin
```

At this point, graphs should start being generated for disk, network, processes, system, etc.

3.18.2 HTTP Traffic

HTTP traffic can be monitored from the client side, the server side, or both.

Monitoring HTTP Traffic from the Client Side

HTTP traffic for web clients that have cookies enabled (most browsers) can be tracked by Google Analytics (https://www.google.com/analytics/) and Matomo (formerly "Piwik"; https://matomo.org/) as explained in the *Web Analytics Code* section of the Installation Guide.

To track analytics beyond pageviews, style classes have been added for end user action buttons, which include:

`btn-compute`, `btn-contact`, `btn-download`, `btn-explore`, `btn-export`, `btn-preview`, `btn-request`, `btn-share`

Monitoring HTTP Traffic from the Server Side

There are a wide variety of solutions available for monitoring HTTP traffic from the server side. The following are merely suggestions and a pull request against what is written here to add additional ideas is certainly welcome! Are you excited about the ELK stack (Elasticsearch, Logstash, and Kibana)? The TICK stack (Telegraph InfluxDB Chronograph and Kapacitor)? GoAccess? Prometheus? Graphite? Splunk? Please consider sharing your work with the Dataverse community!

AWStats

AWStats is a venerable tool for monitoring web traffic based on Apache access logs. On RHEL/CentOS 7, you can try the following steps.

Enable the EPEL yum repo:

```
yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

Install AWStats:

```
yum install awstats
```

Assuming you are using HTTPS rather than HTTP (and you should!), edit `/etc/awstats/awstats.standalone.conf` and change `LogFile="/var/log/httpd/access_log"` to `LogFile="/var/log/httpd/ssl_access_log"`. In the same file, change `LogFormat=1` to `LogFormat=4`. Make both of these changes (`LogFile` and `LogFormat` in `/etc/awstats/awstats.localhost.localdomain.conf` as well.

Process the logs:

```
/usr/share/awstats/tools/awstats_updateall.pl now
```

Please note that load balancers (such as Amazon's ELB) might interfere with the `LogFormat` mentioned above. To start troubleshooting errors such as `AWStats did not find any valid log lines that match your LogFormat parameter`, you might need to bump up the value of `NbOfLinesForCorruptedLog` in the config files above and re-try while you interate on your Apache and AWStats config.

Please note that the Dataverse Project team has attempted to parse Glassfish/Payara logs using AWStats but it didn't seem to just work and posts have been made at https://stackoverflow.com/questions/49134154/what-logformat-definition-does-awstats-require-to-parse-glassfish-http-access-logs and https://sourceforge.net/p/awstats/discussion/43428/thread/9b1befda/ that can be followed up on some day.

3.18.3 Database Connection Pool Used by App Server

https://github.com/IQSS/dataverse/issues/2595 contains some information on enabling monitoring of app servers, which is disabled by default. It's a TODO to document what to do here if there is sufficient interest.

3.18.4 actionlogrecord

There is a database table called `actionlogrecord` that captures events that may be of interest. See https://github.com/IQSS/dataverse/issues/2729 for more discussion around this table.

An Important Note about ActionLogRecord Table:

Please note that in a busy production installation this table will be growing constantly. See the note on *How to Keep ActionLogRecord in Trim* in the Troubleshooting section of the guide.

3.18.5 Edit Draft Versions Logging

Changes made to draft versions of datasets are logged in a folder called logs/edit-drafts. See https://github.com/IQSS/dataverse/issues/5145 for more information on this logging.

3.18.6 Solr Indexing Failures Logging

Failures occurring during the indexing of Dataverse collections and datasets are logged in a folder called logs/process-failures. This logging will include instructions for manually re-running the failed processes. It may be advantageous to set up a automatic job to monitor new entries into this log folder so that indexes could be re-run.

3.18.7 EJB Timers

Should you be interested in monitoring the EJB timers, this script may be used as an example:

```
#!/usr/bin/env bash

# example monitoring script for EBJ timers.
# currently assumes that there are two timers
# real monitoring commands should replace the echo statements for production use

r0=`curl -s http://localhost:8080/ejb-timer-service-app/timer`

if [ $? -ne 0 ]; then
        echo "alert - no timer service" # put real alert command here
fi

r1=`echo $r0 | grep -c "There are 2 active persistent timers on this container"`

if [ "1" -ne "$r1" ]; then
        echo "alert - no active timers" # put real alert command here
fi
```

3.18.8 AWS RDS

Some installations of Dataverse use AWS's "database as a service" offering called RDS (Relational Database Service) so it's worth mentioning some monitoring tips here.

There are two documents that are especially worth reviewing:

- Monitoring an Amazon RDS DB instance: The official documentation.
- Performance Monitoring Workshop for RDS PostgreSQL and Aurora PostgreSQL: A workshop that steps through practical examples and even includes labs featuring tools to generate load.

Tips:

- Enable **Performance Insights**. The product page includes a video from 2017 that is still compelling today. For example, the Top SQL tab shows the SQL queries that are contributing the most to database load. There's also a video from 2018 mentioned in the overview that's worth watching.
  - Note that Performance Insights is only available for PostgreSQL 10 and higher (also mentioned in docs). Version 11 has digest statistics enabled automatically but there's an extra step for version 10.
  - Performance Insights policies describes how to give access to Performance Insights to someone who doesn't have full access to RDS (`AmazonRDSFullAccess`).
- Enable the **slow query log** and consider using pgbadger to analyze the log files. Set `log_min_duration_statement` to "5000", for example, to log all queries that take 5 seconds or more. See enable query logging in the user guide or slides from the workshop for details. Using pgbadger is also mentioned as a common DBA task.
- Use **CloudWatch**. CloudWatch gathers metrics about CPU utilization from the hypervisor for a DB instance. It's a separate service to log into so access can be granted more freely than to RDS. See CloudWatch docs.
- Use **Enhanced Monitoring**. Enhanced Monitoring gathers its metrics from an agent on the instance. See Enhanced Monitoring docs.
- It's possible to view and act on **RDS Events** such as snapshots, parameter changes, etc. See Working with Amazon RDS events for details.
- RDS monitoring is available via API and the `aws` command line tool. For example, see Retrieving metrics with the Performance Insights API.
- To play with monitoring RDS using a server configured by dataverse-ansible set `use_rds` to true to skip some steps that aren't necessary when using RDS. See also the *Deployment* section of the Developer Guide.

3.18.9 MicroProfile Metrics endpoint

Payara provides the metrics endpoint: <https://docs.payara.fish/community/docs/Technical%20Documentation/MicroProfile/Metrics/Metrics%20Rest%20Endpoint.html>_ The metrics you can retrieve that way: - *index_permit_wait_time_seconds_mean* displays how long does it take to receive a permit to index a dataset. - *index_time_seconds* displays how long does it take to index a dataset.

3.19 Rate Limiting

Contents:

- *Configuration*
- *Statistics*

3.19.1 Configuration

Rate limiting is used to prevent users from over taxing the system either deliberately or by runaway automated processes. Rate limiting can be configured on a tier level with tier 0 being reserved for guest users and tiers 1-any for authenticated users. New users are defaulted to tier 1. Superuser accounts are exempt from rate limiting. Rate limits can be imposed on command APIs by configuring the tier, the command, and the hourly limit in the database. Two database settings configure the rate limiting. Note: If either of these settings exist in the database rate limiting will be enabled (note that a Payara restart is required for the :RateLimitingDefaultCapacityTiers setting to take effect). If neither setting exists rate limiting is disabled.

- :RateLimitingDefaultCapacityTiers is the number of calls allowed per hour if the specific command is not configured. The values represent the number of calls per hour per user for tiers 0,1,... A value of -1 can be used to signify no rate limit. Tiers not specified in this setting will default to *-1* (No Limit). I.e., -d "10000" is equivalent to -d "10000,-1,-1,..."

```
curl http://localhost:8080/api/admin/settings/:RateLimitingDefaultCapacityTiers -X PUT -
↪d '10000,20000'
```

- :RateLimitingCapacityByTierAndAction is a JSON object specifying the rate by tier and a list of actions (commands). This allows for more control over the rate limit of individual API command calls. In the following example, calls made by a guest user (tier 0) for API GetLatestPublishedDatasetVersionCommand is further limited to only 10 calls per hour, while an authenticated user (tier 1) will be able to make 30 calls per hour to the same API.

`rate-limit-actions.json` Example JSON for RateLimitingCapacityByTierAndAction

```
curl http://localhost:8080/api/admin/settings/:RateLimitingCapacityByTierAndAction -X␣
↪PUT -d '[{"tier": 0, "limitPerHour": 10, "actions": [
↪"GetLatestPublishedDatasetVersionCommand", "GetPrivateUrlCommand", "GetDatasetCommand",
↪ "GetLatestAccessibleDatasetVersionCommand"]}, {"tier": 0, "limitPerHour": 1, "actions
↪": ["CreateGuestbookResponseCommand", "UpdateDatasetVersionCommand",
↪"DestroyDatasetCommand", "DeleteDataFileCommand", "FinalizeDatasetPublicationCommand",
↪"PublishDatasetCommand"]}, {"tier": 1, "limitPerHour": 30, "actions": [
↪"CreateGuestbookResponseCommand", "GetLatestPublishedDatasetVersionCommand",
↪"GetPrivateUrlCommand", "GetDatasetCommand", "GetLatestAccessibleDatasetVersionCommand
↪", "UpdateDatasetVersionCommand", "DestroyDatasetCommand", "DeleteDataFileCommand",
↪"FinalizeDatasetPublicationCommand", "PublishDatasetCommand"]}]'
```

3.19.2 Statistics

In order to monitor the rate limiting cache for investigative purposes there is a stats endpoint which returns CSV formatted text.

The CSV contains multiple lists.

- The first list contains the username:command, and number of tokens remaining. This list is sorted by the values and has the header "#<username>:<command>, <available tokens>".
- The second list contains username:command, last updated timestamp in minutes, and delta minutes before now. The header for this list is "#<username>:<command>, <timestamp>, <delta minutes> ## deltaMinutesFilter=1". This list can be filtered to show only the entries with updates within the deltaMinutesFilter requested. ("## deltaMinutesFilter=n" will be added to the header when the filter is included in the call.

```
curl http://localhost:8080/api/admin/rateLimitStats
curl http://localhost:8080/api/admin/rateLimitStats?deltaMinutesFilter=10
```

3.20 Reporting Tools and Common Queries

Reporting tools and queries created by members of the Dataverse community.

- Matrix (<https://github.com/rindataverse/matrix>): Collaboration Matrix is a visualization showing the connectedness and collaboration between authors and their affiliations. Visit https://rin.lipi.go.id/matrix/ to play with a production installation.
- Dataverse Installation Web Report (<https://github.com/scholarsportal/Dataverse-Web-Report>): Creates interactive charts showing data extracted from the Dataverse installation Excel Report
- Dataverse Installation Reports for Texas Digital Library (<https://github.com/TexasDigitalLibrary/dataverse-reports>): A python3-based tool to generate and email statistical reports from Dataverse (https://dataverse.org/) using the native API and database queries.
- dataverse-metrics (<https://github.com/IQSS/dataverse-metrics>): Aggregates and visualizes metrics for Dataverse installations around the world or a single Dataverse installation.
- Automated Dataverse Metrics Reports (<https://github.com/QualitativeDataRepository/dataverse-metrics>): Create beautiful, customizable reports about your Dataverse installation at the click of a button, using R-Markdown and GitHub Actions.
- dataverse-metadata-crawler (https://github.com/scholarsportal/dataverse-metadata-crawler): A Python CLI tool for extracting and exporting metadata from Dataverse repositories. It supports bulk extraction of dataverses, datasets, and data file metadata from any chosen level of dataverse collection (an entire Dataverse repository/sub-Dataverse), with flexible export options to JSON and CSV formats.
- Useful queries from the Dataverse Community (<https://docs.google.com/document/d/1-Y_iUduSxdDNeK1yiGUxe7t-Md7Fy965jp4o4m1XEoE/edit#heading=h.avuoo5kf0mdt>): A community-generated and maintained document of postgresql queries for getting information about users and dataverse collections, datasets, and files in your Dataverse installation. If you are trying to find out some information from your Dataverse installation, chances are that someone else has had the same questions and it's now listed in this document. If it's not listed, please feel free to add it to the document.

3.21 Maintenance

When you have scheduled down time for your production servers, we provide a `sample maintenance page` for you to use. To download, right-click and select "Save Link As".

The maintenance page is intended to be a static page served by Apache to provide users with a nicer, more informative experience when the site is unavailable.

3.22 Backups

Running tape, or similar backups to ensure the long term preservation of all the data stored in the Dataverse Repository is an implied responsibility that should be taken most seriously.

In addition to running these disk-level backups, we have provided an experimental script that can be run on schedule (via a cron job or something similar) to create extra archival copies of all the Datafiles stored in the Dataverse Repository on a remote storage server, accessible via an ssh connection. The script and some documentation can be found in `scripts/backup/run_backup` in the Dataverse Software source tree at https://github.com/IQSS/dataverse . Some degree of knowledge of system administration and Python is required.

Once again, the script is experimental and NOT a replacement of regular and reliable disk backups!

3.23 Troubleshooting

Sometimes a Dataverse installation's users get into trouble. Sometimes a Dataverse installation itself gets into trouble. If something has gone wrong, this section is for you.

Contents:

- *Using Dataverse Installation APIs to Troubleshoot and Fix Problems*
  - *A Dataset Is Locked And Cannot Be Edited or Published*
  - *Someone Created Spam Datasets and I Need to Delete Them*
  - *A User Needs Their Account to Be Converted From Institutional (Shibboleth), ORCID, Google, or GitHub to Something Else*
- *Ingest*
  - *Long-Running Ingest Jobs Have Exhausted System Resources*
- *Payara*
  - *Finding the Payara Log File*
  - *Increasing Payara Logging*
- *Deployment fails, "EJB Timer Service not available"*
- *Timer Not Working*
- *Constraint Violations Issues*
- *Many Files with a File Type of "Unknown", "Application", or "Binary"*
- *What's with this Table "ActionLogRecord" in Our Database, It Seems to be Growing Uncontrollably?*
- *Getting Help*

3.23.1 Using Dataverse Installation APIs to Troubleshoot and Fix Problems

See the *Introduction* section of the API Guide for a high level overview of Dataverse Software APIs. Below are listed problems that support teams might encounter that can be handled via API (sometimes only via API).

A Dataset Is Locked And Cannot Be Edited or Published

There are several types of dataset locks. Locks can be managed using the locks API, or by accessing them directly in the database. Internally locks are maintained in the `datasetlock` database table, with the field `dataset_id` linking them to specific datasets, and the column `reason` specifying the type of lock.

It's normal for the ingest process described in the *Tabular Data, Representation, Storage and Ingest* section of the User Guide to take some time but if hours or days have passed and the dataset is still locked, you might want to inspect the locks and consider deleting some or all of them. It is recommended to restart the application server if you are deleting an ingest lock, to make sure the ingest job is no longer running in the background. Ingest locks are idetified by the label `Ingest` in the `reason` column of the `DatasetLock` table in the database.

A dataset is locked with a lock of type `finalizePublication` while the persistent identifiers for the datafiles in the dataset are registered or updated, and/or while the physical files are being validated by recalculating the checksums and

verifying them against the values stored in the database, before the publication process can be completed (Note that either of the two tasks can be disabled via database options - see *Configuration*). If a dataset has been in this state for a long period of time, for hours or longer, it is somewhat safe to assume that it is stuck (for example, the process may have been interrupted by an application server restart, or a system crash), so you may want to remove the lock (to be safe, do restart the application server, to ensure that the job is no longer running in the background) and advise the user to try publishing again. See *Managing Datasets and Dataverse Collections* for more information on publishing.

If any files in the dataset fail the validation above the dataset will be left locked with a `DatasetLock.Reason=FileValidationFailed`. The user will be notified that they need to contact their Dataverse installation's support in order to address the issue before another attempt to publish can be made. The admin will have to address and fix the underlying problems (by either restoring the missing or corrupted files, or by purging the affected files from the dataset) before deleting the lock and advising the user to try to publish again. The goal of the validation framework is to catch these types of conditions while the dataset is still in DRAFT.

During an attempt to publish a dataset, the validation will stop after encountering the first file that fails it. It is strongly recommended for the admin to review and verify *all* the files in the dataset, so that all the compromised files can be fixed before the lock is removed. We recommend using the `/api/validate/dataset/files/{id}` API. It will go through all the files for the dataset specified, and will report which ones have failed validation. see *Physical Files Validation in a Dataset* in the *Native API* section of the User Guide.

The following are two real life examples of problems that have resulted in corrupted datafiles during normal operation of a Dataverse installation:

1. Botched file deletes - while a datafile is in DRAFT, attempting to delete it from the dataset involves deleting both the `DataFile` database table entry, and the physical file. (Deleting a datafile from a *published* version merely removes it from the future versions - but keeps the file in the dataset). The problem we've observed in the early versions of the Dataverse Software was a *partially successful* delete, where the database transaction would fail (for whatever reason), but only after the physical file had already been deleted from the filesystem. Thus resulting in a datafile entry remaining in the dataset, but with the corresponding physical file missing. We believe we have addressed the issue that was making this condition possible, so it shouldn't happen again - but there may be a datafile in this state in your database. Assuming the user's intent was in fact to delete the file, the easiest solution is simply to confirm it and purge the datafile entity from the database. Otherwise the file needs to be restored from backups, or obtained from the user and copied back into storage.

2. Another issue we've observed: a failed tabular data ingest that leaves the datafile un-ingested, BUT with the physical file already replaced by the generated tab-delimited version of the data. This datafile will fail the validation because the checksum in the database matches the file in the original format (Stata, SPSS, etc.) as uploaded by the user. To fix: luckily, this is easily reversible, since the uploaded original should be saved in your storage, with the .orig extension. Simply swapping the .orig copy with the main file associated with the datafile will fix it. Similarly, we believe this condition should not happen again in Dataverse Software 4.20+, but you may have some legacy cases on your server.

Someone Created Spam Datasets and I Need to Delete Them

Depending on how open your Dataverse installation is to the general public creating datasets, you may sometimes need to deal with spam datasets.

Look for "destroy" in the *Native API* section of the API Guide.

A User Needs Their Account to Be Converted From Institutional (Shibboleth), ORCID, Google, or GitHub to Something Else

See *Converting Shibboleth Users to Local* and *Converting OAuth Users to Local*.

3.23.2 Ingest

Long-Running Ingest Jobs Have Exhausted System Resources

Ingest is both CPU- and memory-intensive, and depending on your system resources and the size and format of tabular data files uploaded, may render your Dataverse installation unresponsive or nearly inoperable. It is possible to cancel these jobs by purging the ingest queue.

`/usr/local/payara7/mq/bin/imqcmd -u admin query dst -t q -n DataverseIngest` will query the DataverseIngest destination. The password, unless you have changed it, matches the username.

`/usr/local/payara7/mq/bin/imqcmd -u admin purge dst -t q -n DataverseIngest` will purge the DataverseIngest queue, and prompt for your confirmation.

Finally, list destinations to verify that the purge was successful:

```
/usr/local/payara7/mq/bin/imqcmd -u admin list dst
```

3.23.3 Payara

Finding the Payara Log File

`/usr/local/payara7/glassfish/domains/domain1/logs/server.log` is the main place to look when you encounter problems (assuming you installed Payara in the default directory). Hopefully an error message has been logged. If there's a stack trace, it may be of interest to developers, especially they can trace line numbers back to a tagged version or commit. Send more of the stack trace (the entire file if possible) to developers who can help (see "Getting Help", below) and be sure to say which version of the Dataverse Software you have installed.

Increasing Payara Logging

For debugging purposes, you may find it helpful to temporarily increase logging levels. Here's an example of increasing logging for the Java class behind the "datasets" API endpoints:

```
./asadmin set-log-levels edu.harvard.iq.dataverse.api.Datasets=FINE
```

For more on setting log levels, see the *Debugging* section of the Developer Guide.

Our guides focus on using the command line to manage Payara but you might be interested in an admin GUI at http://localhost:4848

3.23.4 Deployment fails, "EJB Timer Service not available"

Sometimes your Dataverse installation fails to deploy, or Payara fails to restart once the application is deployed, with the following error message: "remote failure: Error occurred during deployment: Exception while loading the app : EJB Timer Service is not available. Please see server.log for more details."

We don't know what's causing this issue, but here's a known workaround:

- Stop Payara;
- Remove the `generated` and `osgi-cache` directories from the `domain1` directory;
- Start Payara

The shell script below performs the steps above. Note that it may or may not work on your system, so it is provided as an example only, downloadable `here`. The configuration values might need to be changed to reflect your environment (the Payara directory). See the comments in the script for more information.

```
#!/bin/sh

# EBJ timers sometimes cause problems; utility to clear generated directories

# assumes this script is being run as root

# will restart Payara if it's stopped; comment out the `start-domain` command at the end
# if you'd like to avoid that.

# directory where Payara is installed
PAYARA_DIR=/usr/local/payara7

# directory within Payara (defaults)
DV_DIR=${PAYARA_DIR}/glassfish/domains/domain1

# stop the domain (generates a warning if app server is stopped)
${PAYARA_DIR}/bin/asadmin stop-domain

rm -rf ${PAYARA_DIR}/${DV_DIR}/generated/
rm -rf ${PAYARA_DIR}/${DV_DIR}/osgi-cache/

# restart the domain (also generates a warning if app server is stopped)
${PAYARA_DIR}/bin/asadmin start-domain
```

3.23.5 Timer Not Working

Your Dataverse installation relies on EJB timers to perform scheduled tasks: harvesting from remote servers, updating the local OAI sets and running metadata exports. (See *Dataverse Installation Application Timers* for details.) If these scheduled jobs are not running on your server, you might experience the following symptoms:

If you are seeing the following in your server.log…

Handling timeout on …

followed by an Exception stack trace with these lines in it:

Internal Exception: java.io.StreamCorruptedException: invalid stream header …

Exception Description: Could not deserialize object from byte array …

... you should reach out by opening an issue. In the good ol' days of running Dataverse Software 4.x running on Glassfish 4, this was a hint for an unsupported JDBC driver. In Dataverse Software 5.x this would be a new regression and its cause would need to be investigated.

3.23.6 Constraint Violations Issues

In real life production use, it may be possible to end up in a situation where some values associated with the datasets in your database are no longer valid under the constraints enforced by the later versions of the Dataverse Software. This is not very likely to happen, but if it does, the symptoms will be as follows: Some datasets can no longer be edited, long exception stack traces logged in the app server log, caused by:

```
javax.validation.ConstraintViolationException:
Bean Validation constraint(s) violated while executing Automatic Bean Validation on␣
↪callback event:'preUpdate'.
Please refer to embedded ConstraintViolations for details.
```

(contrary to what the message suggests, there are no specific "details" anywhere in the stack trace that would explain what values violate which constraints)

To identify the specific invalid values in the affected datasets, or to check all the datasets in the Dataverse installation for constraint violations, see *Dataset Validation* in the *Native API* section of the User Guide.

3.23.7 Many Files with a File Type of "Unknown", "Application", or "Binary"

From the home page of a Dataverse installation you can get a count of files by file type by clicking "Files" and then scrolling down to "File Type". If you see a lot of files that are "Unknown", "Application", or "Binary" you can have the Dataverse installation attempt to redetect the file type by using the *Redetect File Type* API endpoint.

3.23.8 What's with this Table "ActionLogRecord" in Our Database, It Seems to be Growing Uncontrollably?

An entry is created in ActionLogRecord table every time an application command is executed (to be precise, certain non-command actions, such as logins are recorded there as well). This is very useful for investigating problems or usage patterns. However, please note that there is no builtin mechanism in the Application for trimming this table, so it will continue growing as your Dataverse installation is kept in operation. For example, multiple entries in this table are created every time a guest user views the page of a published dataset. Many more are created when an author is actively working on a dataset, making edits, adding new files, etc. On a busy installation this table is likely to grow at a faster rate than the actual data holdings. For example, after five years of production use at Harvard IQSS, the raw size of ActionLogRecord appeared to exceed the combined size of the rest of the database (!). It's worth pointing out that the sheer size of this one table does not by itself result in performance issues in any linear way. But it may still be undesirable to keep that much extra data around; especially since for most installations these records are unlikely to have much value past a certain number of months or years. Some installations may be purchasing their database services from cloud computing providers (RDS, etc.) where extra data may result in higher costs. Here at Harvard we chose to periodically trim the table manually, deleting all the entries older than 2 years. We recommend that you check on the size of this table in your database, and choose whether, and how often you want to trim it. You will also need to decide whether you want to archive these older records outside the database before deleting them. If you see no reason to keep them around, older records can be erased with a simple query. For example, to delete everything before the year 2021:

```
DELETE FROM ACTIONLOGRECORD WHERE starttime < '2021-01-01 00:00:00';
```

If you want to preserve these old entries before deleting them, you can save them with, for example, psql:

```
psql <CREDENTIALS> -d <DATABASE_NAME> -t -c "SELECT * FROM actionlogrecord WHERE
starttime < '2021-01-01 00:00:00' ORDER BY starttime;"
```

A full backup of the table can be made with pg_dump, for example:

```
pg_dump <CREDENTIALS> --table=actionlogrecord --data-only <DATABASE_NAME> > /tmp/
actionlogrecord_backup.sql
```

In the example above, the output will be saved in raw SQL format. It is portable and human-readable, but uses a lot of space. It does, however, compress very well.

Add the `-Fc` option to save the output in a proprietary, compressed binary format which will dump and restore much more quickly, but note that in this format dead tuples will be copied as well. To reduce the amount of storage consumed by the newly-trimmed `actionlogrecord` table, you must issue `vacuum full actionlogrecord` before dumping the database in this custom format.

3.23.9 Getting Help

If the troubleshooting advice above didn't help, contact any of the support channels mentioned in the *Getting Help* section of the Installation Guide.

3.24 Scaling Dataverse with Data Size

This section is intended to help administrators configure Dataverse appropriately to handle larger amounts of data.

Scaling is a complex subject: there are many options available in Dataverse that can improve performance with larger scale data, some of which work differently than Dataverse's default configuration, potentially requiring user education, and some which can require additional expertise to manage.

In general, there are three dimensions in which Dataverse can scale:

1. **Storage size of individual files and aggregate storage size**
2. **Number of files per dataset**
3. **Number of datasets**

Contents:

- *Storage: Choosing the Right Store*
  - *File Stores*
  - *S3 Stores: Object Storage via S3*
    - *Other Considerations*
  - *Remote Stores*
  - *Globus Stores: Globus Transfer Between Large File/Tape Archives*
- *Storage Strategy Recommendations*
- *Managing More Files per Dataset and More Datasets*
  - *Avoiding Many Files*
  - *Scaling-related Configuration*

- *Scaling Infrastructure*

3.24.1 Storage: Choosing the Right Store

The main issues for handling larger files and larger aggregate data size relate to the performance of the storage used and how involved the Dataverse server is in the data transfer. With appropriate configuration, Dataverse can support file sizes and aggregate dataset sizes into the terabyte scale and beyond.

The primary choice in Dataverse related to storage is which types of "store" (also called "storage driver") to use:

File Stores

The default storage option in Dataverse uses the local file system. When files are transferred to Dataverse, they are first stored in a temporary location on the Dataverse server. Any zip files uploaded are unzipped to create multiple individual file entries. Once an upload is completed, Dataverse copies the files to permanent storage. Dataverse also takes advantage of the file being local to inspect its bytes to determine its MIME type, and, for tabular data, to "*ingest*" it - extracting metadata about the variables used in the file and creating a tab-separated values (TSV) version of the file.

Benefits:

- This option requires no external services and can potentially handle files into the gigabyte (GB) size range. For smaller institutions, and in disciplines where datasets do not have more than a few hundred files and files are not too large, this can be the simplest option.
- Unzipping of zip archives can be simpler for users than having to upload many individual files and was, at one time, the only way to preserve file path names when uploading. In addition, some of the unzipped files might be in a format that can be previewed or otherwise acted upon - see *File Handling* in the User Guide.

Challenges: In general, file storage is not a good option for larger data sizes - both in terms of file size and number of files. Contributing factors include:

- Because temporary storage is used, transfers will temporarily use several times as much space as the final transfer. Unzipping also increases the final storage size of a dataset.
- Because all uploads use the same temporary storage, temporary storage must be large enough to handle multiple users uploading data.
- Each file is uploaded as a single HTTP request, which can cause long transfer times which, in turn, can trigger timeout errors in Dataverse or any proxy or load balancer in front of Dataverse.
- Uploading many files at once can trigger any rate limiter in front of the Dataverse server (i.e. used to throttle AI bots) resulting in failures.
- Because transfers (both uploads and downloads) are handled by the Dataverse server, they add to server processing load which can affect overall performance.
- Cost: local file storage must be provisioned in advance based on anticipated demand. It can involve up-front costs (for a local disk), or, when procured from a cloud provider, is likely to be more expensive than object storage from that provider (see below).

S3 Stores: Object Storage via S3

A more scalable option for storage is to use an object store with transfers managed using the Simple Storage Service (S3) protocol. S3-compatible storage can easily be bought (rented) from major cloud providers, but may also be available from institutional clouds. It is also possible to run open source software to provide S3 storage over a local file system (making it possible to enjoy the advantages discussed below while still leveraging local file storage).

While S3 Stores can be configured to handle uploads and downloads as with file storage (with zip files being unzipped, but having many of the same challenges in terms of temporary storage and server load as discussed above) they can also be configured to use "direct" upload and download. In this configuration, the actual transfer of file bytes is between the user's local machine and the S3 store. In this configuration, files are never stored on the Dataverse server. Dataverse does not attempt to unzip zip files, and they are stored as a single file in the dataset.

Benefits: S3 offers several advantages over file storage:

- Scalability: S3 is designed to handle large amounts of data. It can handle individual files up to several TB in size. Because S3 supports breaking files into pieces, Dataverse can transfer a file in pieces (several in parallel, potentially thousands of pieces per file) making transfers faster and more robust (a failure requires only resending the failed piece). It may also be the case that users will have a faster network connection to the S3 store (e.g. in a commercial cloud or High Performance Computing center) than they do to the Dataverse server, reducing transfer time.
- High Availability: S3 provides redundancy beyond what is available with a local file system (valuable for preservation, potentially reducing the need to perform data integrity checks).

Challenges:

- One additional step that is required to enable direct uploads via a Dataverse installation and for direct download to work with previewers and direct upload to work with DVWebloader (*Folder Upload*) is to allow *cross site (CORS) requests on your S3 store*.
- Cost: S3 offers a pricing model that allows you to pay for the storage and transfer of data based on current usage (versus long term demand) but commercial providers charge more per TB than the equivalent cost of a local disk (though commercial S3 storage is cheaper than commercial file storage). There can also be egress and other charges. Overall, S3 storage is generally more expensive than local file storage but cheaper than cloud file storage. Running a local S3 storage or leveraging an institutional service can further reduce costs.
- Direct upload via S3 is a *multi-step process*: Dataverse provides URLs for the uploads, the user's browser or other app uses the URLs to transfer files to the S3 store, possibly in many pieces per file, and finally, Dataverse is told that one or more files are in place and should be added to the dataset. If the last step fails, or if all parts of a file cannot be transferred, orphaned files or parts of files can be left on S3. These files are not accessible via Dataverse but do use space (for which there is a monetary cost) until they are deleted. There is currently no automated clean-up mechanism.

Other Considerations

- S3 Storage without direct upload/download provides minimal benefits with Dataverse as files still pass through the server, files are still uploaded as a single HTTP/HTTPS stream, and temporary storage is still used.
- While not having files unzipped can be confusing to users who are used to it from using Dataverse with file storage, there are ways to minimize the impact. For example, Dataverse can be configured to use a "Zip File Previewer" that allows users to see the contents of a zip file and even download individual files from within it (see *Compressed Files*). For users who still want their data stored as individual files with their relative folder paths, Dataverse can be configured with "*DVWebloader*" which allows users to select an entire folder tree of files and upload them, with their relative paths intact, to Dataverse. (DVWebloader can only be used with S3/direct upload, but it is much more efficient with many files than using the standard upload interface in Dataverse (which also does not retain path information)).

- Several features that involve Dataverse accessing files' contents, including unzipping zip files, are disabled when S3 direct upload is enabled. See *Features that are Disabled if S3 Direct Upload is Enabled*.
- Using direct upload stops Dataverse from inspecting the file bytes to determine the MIME type (with one exception - Stata files). Dataverse will still look at the file name and extension to determine the MIME type.
- To perform "ingest" processing (see *Tabular Data File Ingest*), Dataverse currently has to copy the file to local storage, negating the benefit of sending data directly to S3. To manage larger files, one can set a per-store ingest size limit (which can be 0 bytes) to stop ingest or limit it to smaller files (see *List of S3 Storage Options*).
- Dataverse's mechanism for downloading a whole dataset or multiple selected files involves zipping those files together. Even When using S3 with direct upload/download, the file bytes are transferred to the Dataverse server as part of the zipping process. There are ways to reduce the performance impact of this:
  - There is a *Standalone "Zipper" Service Tool* that can be run separate from Dataverse to handle the zipping process.
  - Dataverse has a *:ZipDownloadLimit* that can be used to limit the amount of data that can be zipped. If a dataset is larger than this limit, Dataverse will only add some of the files to the zip and list others in the included manifest file.
  - There are tools such as the Dataverse Dataset Downloader (https://github.com/gdcc/dataverse-recipes/tree/main/shell/download#dataverse-dataset-downloader) that can be used to download all of the files individually. This avoids sending any of the files through the Dataverse server when S3 direct download is enabled.
- Dataverse leverages S3 features that are not implemented by all servers and has several configuration options geared towards handling variations between servers - see *Reported Working S3-Compatible Storage*. Site admins should be sure to test with their preferred S3 implementation (and consider adding to the list of working S3 implementations).
- The part-size used when directly transferring files to S3 is configurable (at AWS, from 5 MiB to 5GiB). The default in Dataverse is 1 GiB (1024^3 bytes). If the primary use case is with smaller files than that, decreasing the part size may improve upload speeds.

Remote Stores

Note: Remote Storage is still experimental: feedback is welcome! See *Getting Help*.

For very large, and/or very sensitive data, it may not make sense to transfer or copy files to Dataverse at all. The `remote` store type in the Dataverse software supports these use cases. It allows Dataverse to store a URL reference to the file rather than transferring the file bytes to a store managed directly by Dataverse. In the most basic configuration a site administrator configures the base URL for the store, e.g. "https://thirdpartystorage.edu/long-term-storage/" and users can then create files referencing any URL starting with that base, e.g. "https://thirdpartystorage.edu/long-term-storage/my_project_dir/my_file.txt".

If the remote site is a public web server, the remote store in Dataverse should be configured to be "public" which will disable the ability to restrict or embargo files (as they are public on the remote site and Dataverse cannot block access.) Conversely, Dataverse can be configured to sign requests to the remote server which the remote server can then, if it is capable of validating them, use to reject requests not approved by Dataverse. In this configuration, users can restrict and embargo files and Dataverse and the remote server will cooperate to manage access control. Another alternative, with a more advanced remote store, would be, instead of using URLs that directly enable download of the file, to use URLs that point to a landing page at the remote server that may require the user to login and go through some approval process before being able to access the file.

Dataverse considers remote storage to be read-only, or, in cases where the remote service does not provide a way for Dataverse to download the file bytes (due to access control or because the URL refers to a landing page), inaccessible. Depending on whether Dataverse can access the bytes of the file, functionality such as ingest and integrity checking may or may not be possible. If the file bytes are not accessible, the remote store in Dataverse should be configured to

disable operations that attempt to access the file (see the files-not-accessible-by-dataverse in *Trusted Remote Storage*). Regardless of whether the remote files can be read, local storage of other datafiles and auxiliary files in the same dataset is possible. Support for such files is handled by configuring a "base" store with the remote store that is used for these purposes. (This means that while files added as remote remain on the remote store, other files in the dataset, and potentially thumbnails and the ingested TSV format of remote files would be managed by Dataverse in the base store. If ingest is not desired, the ingest size limit for the store can be set to 0 bytes).

Benefits:

- This is a relatively simple way to off-load the management of large and/or sensitive data files to other organizations while still providing Dataverse's overall capabilities for dataset curation and publication to users.
- If the store has been configured with a remote-store-name or remote-store-url, the dataset file table will include this information for remote files. These provide a visual indicator that the files are not managed directly by Dataverse and are stored/managed by a remote trusted store.

Challenges:

- As Dataverse is relying on the remote service to maintain the integrity and availability of the files, it is likely that the Dataverse site admin will want to have a formal agreement with the remote service operator about their policies.
- Currently, remote files can only be added via the API. (This may be addressed in future versions).
- Remote files can only be added after the dataset is created in the UI (and therefore has an id and PID for use with the API). However, the UI will still allow upload of files to the base store (at dataset creation and when editing), which could be confusing.
- Site admins need to consider carefully how to configure file size limits, ingest size limits, etc. on the remote store and its base store, and whether the remote store is public-only, and whether files there can be read by Dataverse to assure the requirements of a specific use case(s) are addressed.
- The current remote store implementation will not prevent you from providing a relative URL that results in a 404 when resolved (i.e. if you make a typo). You should check to make sure the file exists at the location you specify - by trying to download in Dataverse, by checking to see that Dataverse was able to get the file size (which it does with a HEAD call to that location), or just manually trying the URL in your browser.
- For large files, direct-download should always be used with a remote store. (Otherwise the Dataverse will be involved in the download.)
- When multiple files are selected for download, Dataverse will try to include remote files in the zip file being created (up to the max zip size limit) which is inefficient, and will not be able to include remote files that are inaccessible (possibly confusing).

Globus Stores: Globus Transfer Between Large File/Tape Archives

Note: Globus Transfer is still experimental: feedback is welcome! See *Getting Help*.

Globus provides file transfer service that is widely used for the world's largest datasets (in terms of both file size and number of files). It provides:

- Robust file transfer capable of handling delays (e.g. due to the time it takes to mount tapes) and restarting after network or endpoint failures
- Rapid parallel file transfers, potentially between clusters of computers on both ends
- Third-party transfer, which enables a user working with their desktop browser to initiate transfers of files between remote endpoints, i.e. sending files on a local high-performance computing cluster to a Dataverse endpoint or vice versa.

Dataverse can be configured to support Globus transfers in multiple ways:

- A Dataverse-managed Globus File Endpoint: Dataverse controls user access to the endpoint, access is only via Globus
- A Dataverse-managed Globus S3 Endpoint: Dataverse controls user access to the endpoint, access is available via S3 and via Globus
- A Globus Endpoint treated as Remote Storage: Dataverse references files on a Globus endpoint managed by a third party

Benefits:

- Globus scales to higher data volumes than any other option. Users working with large data are often familiar with Globus and are interested in transferring data to/from computational clusters rather than their local machine.
- Globus transfers can be initiated by choosing the Globus option in the dataset upload panel. Analogously, "Globus Transfer" is one of the download options in the "Access Dataset" menu.
- For the non-S3 options, Dataverse supports having a base store (e.g. a local file system or an S3-based store), which can be used internally by Dataverse (e.g. for thumbnails, etc.) and can allow users to upload smaller files (e.g. READMEs, documentation) that might not be suited to a given Globus endpoint (e.g. a tape store).

Challenges:

- Globus is complex to manage and Dataverse installations will need to develop Globus expertise or partner with another organization (i.e. an institutional high-performance computing center) to manage Globus endpoints.
- For users not familiar with Globus, managing transfers can be confusing. For the non-S3 options, users cannot just download files - they must have access to a destination Globus endpoint and have a Globus account. Globus does provide free accounts and a free "Globus Personal Connect" service which installed on any machine to allow transfers to/from it.
- Globus transfers are not enabled at dataset-creation time. Once the draft version is created, users can initiate Globus transfers to upload files from remote endpoints.
- For Dataverse-managed endpoints, a community-developed dataverse-globus app must be installed and configured in the Dataverse instance. This app manages granting and revoking access for users to upload/download files from Dataverse and handles the translation between Dataverse's internal file naming/organization to that seen by the user.
- Users familiar with Globus sometimes expect to be able to find the Dataverse endpoint in Globus' online service and download files from there. Due to the fact that Dataverse is managing permissions and handling file naming, this doesn't work.
- Due to differences between Dataverse's and Globus's access control models, Dataverse cannot enforce per-file access restrictions - restriction can only be done today at the level of providing access to all files in a dataset. Globus stores can be defined as public to disable Dataverse's ability to restrict and embargo files in that store. If the store is configured to support restriction and embargo, Dataverse and its Dataverse-Globus app will limit users to downloading only the files they have been granted access to, but a technically knowledgeable user could access other files in the same dataset if they are give access to one. (Data depositors would need to be aware of this limitation and could be guided to restrict all files/only grant access to all dataset files in Globus as a work-around).
- Dataverse-managed endpoints must be Globus "guest collections" hosted on either a file-system-based endpoint or an S3-based endpoint (the latter requires use of the Globus S3 connector which requires a paid Globus subscription at the host institution). In either case, Dataverse is configured with the Globus credentials of a user account that can manage the endpoint. Users will need their own Globus account, which can be obtained via their institution or directly from Globus (at no cost).
- With the file-system endpoint, Dataverse does not currently have access to the file contents. Thus, functionality related to ingest, previews, fixity hash validation, etc. are not available. (Using the S3-based endpoint, Dataverse

has access via S3 and all functionality normally associated with direct uploads to S3 is available. In this case admins should be sure to set the maximum size for ingest and avoid requiring hash validation at publication, etc.)

- For the reference use case, Dataverse must be configured with a list of allowed endpoint/base paths from which files may be referenced. In this case, since Dataverse is not accessing the remote endpoint itself, it does not need Globus credentials. Users will also need a Globus account in this case, and the remote endpoint must be configured to allow them access, i.e. be publicly readable, or potentially supporting some out-of-band mechanism for access requests (which could be described, for example, in the dataset's Terms of Use and Access).
- As with remote stores, files can only be added in the Globus reference case via the Dataverse API.
- While Globus itself can handle many (millions of) files of any size, Dataverse cannot handle more than thousands of files per dataset (at best) and some Globus endpoints may have limits on file sizes - both maximums and minimums (e.g. for tape storage where small files are inefficient). Users will need to be made aware of these limitations and the possibilities for managing them (e.g. by aggregating multiple files in a single, larger file, or storing smaller files in the base-store via the normal Dataverse upload UI).
- There is currently a bug that won't allow users to transfer files from/to endpoints where they do not have permission to list the overall file tree (i.e. an institution manages <endpoint>/institution_name but the user only has access to <endpoint>/institution_name/my_dir.) Until that is fixed, a work-around is to first transfer data to an endpoint without this restriction.
- An alternative, experimental implementation of Globus polling of ongoing upload transfers was added in v6.4. This framework does not rely on the instance staying up continuously for the duration of the transfer and saves the state information about Globus upload requests in the database. While it is now the recommended option, it is not enabled by default. See the *dataverse.feature.globus-use-experimental-async-framework* feature flag and the JVM option *dataverse.files.globus-monitoring-server*.

More details of the setup required to enable Globus is described in the Community Dataverse-Globus Setup and Configuration document and the references therein.

An overview of the control and data transfer interactions between components was presented at the 2022 Dataverse Community Meeting and can be viewed in the Integrations and Tools Session Video around the 1 hr 28 min mark.

See also *Globus File Transfer* and *Globus settings*.

3.24.2 Storage Strategy Recommendations

Based on both file size and volume considerations, here are some general recommendations:

1. **For research projects with moderate data (< 2GB files, < 100s of files/dataset):**
   - The default File Store is sufficient
   - Consider setting file count and size limits (see below)
2. **For projects with larger files (GBs to TBs) and/or more files per dataset (100s to 1000s):**
   - Configure an S3 store with direct upload/download
   - Set appropriate ingest size limits
3. **For projects with very large files or sensitive data that should remain in place:**
   - Use the Remote Store
4. **For high-performance computing environments or very large datasets (TBs+):**
   - Use a Globus Store
   - Work with users to size files appropriate to the underlying storage

- Consider Globus over S3 when normal upload/download options (via the UI/API) are desired along with Globus transfer

5. **For Petascale datasets, or extreme numbers of files:**
    - Consider a Remote Store and referencing a single URL/Globus endpoint for the entire dataset
6. **For Dataverse installations supporting a range of data scales:**
    - Consider using *multiple stores* and assigning stores to individual collections or datasets

3.24.3 Managing More Files per Dataset and More Datasets

Dataverse can be configured to handle datasets with hundreds or thousands of files and hundreds of thousands of datasets. However, reaching these levels can require significant effort to appropriately configure the server.

Technically, there are two factors that can limit scaling the number of files per dataset: how much the Dataverse server is involved in data transfer, and constraints based on Dataverse's code and database configuration. The former is dramatically affected by the choice for file storage and options such as the S3 direct upload/download settings and ingest size limits. There are fewer ways to affect the latter beyond increasing the amount of memory and CPU resources available or rewriting the relevant parts of Dataverse. (There are continuing efforts to improve Dataverse's performance and scaling, so it is also advisable to use the latest version if you are pushing the boundaries on scaling. Progress is being made.)

Scaling to larger numbers of datasets (and to some extent scaling files per dataset) also depends on Dataverse's Solr search engine. There have been very significant improvements in indexing and search performance in recent releases, including some that are not turned on by default (listed below).

Avoiding Many Files

Before describing things that can be done to improve scaling, it is important to note that there are configuration options and best practices to suggest to users to help avoid larger datasets and help them avoid hitting performance issues by going beyond the file counts you know work in your instance.

There are a number of settings to limit how many files can be uploaded (see *Database Settings* and *JVM Options* for more details):

- *:ZipUploadFilesLimit* - the maximum number of files allowed in an uploaded zip file - only relevant for file stores and S3 when direct upload is not used.
- *:MultipleUploadFilesLimit* - the number of files the GUI user is allowed to upload in one batch, via drag-and-drop, or through the file select dialog
- *:MaxFileUploadSizeInBytes* - limit the size of files that can be uploaded
- *:UseStorageQuotas* - once enabled, super users can set per-collection quotas (in bytes) to limit the aggregate size of all files in the collection
- *dataverse.files.default-dataset-file-count-limit* - directly limits the number of files per dataset, can be changed per dataset via API (by super users)

Scaling-related Configuration

There are a broad range of options (that are not turned on by default) for improving how well Solr indexing and searching scales and for handling more files per dataset. Some of these are useful for all installations while others are related to specific use cases, or are mostly for emergency use (e.g. disabling facets). (see *Database Settings*, *JVM Options*, and *Feature Flags* for more details):

- *dataverse.feature.add-publicobject-solr-field* =true - specifically marks unrestricted content as public in Solr.
- *dataverse.feature.avoid-expensive-solr-join* =true - this tells Dataverse to use the feature above to speed up searches.
- *dataverse.feature.reduce-solr-deletes* =true - when Solr entries are being updated, this avoids an unnecessary step (deletion of existing entries) for entries that are being replaced.
- *dataverse.feature.disable-dataset-thumbnail-autoselect* =true - by default, Dataverse scans through all files in a dataset to find one that can be used as a thumbnail, which is expensive for many files. This disables that behavior to improve performance.
- *dataverse.feature.only-update-datacite-when-needed* =true - reduces the load on DataCite and reduces Dataverse failures related to that load, which is important when using file PIDs on Datasets with many files.
- *dataverse.solr.min-files-to-use-proxy* =<X> - improve performance/lower memory requirements when indexing datasets with many files, suggested value is in the range 200 to 500
- *dataverse.solr.concurrency.max-async-indexes* =<X> - limits the number of index operations running in parallel. The default is 4, larger values may improve performance (if the Solr instance is appropriately sized)
- *:SolrFullTextIndexing* - false improves performance at the expense of not indexing file contents
- *:SolrMaxFileSizeForFullTextIndexing* - size in bytes (default unset/no limit) above which file contents should not be indexed
- *:ZipDownloadLimit* - the maximum size in bytes for zipped downloads of files from a dataset. If the size of requested files is larger, some files will be omitted and listed in the zip manifest file as not included.
- *:DatasetChecksumValidationSizeLimit* - by default, Dataverse checks fixity (assuring the file contents match the recorded checksum) as part of publication. This setting specifies a maximum aggregate dataset size, above which this validation will not be done.
- *:DataFileChecksumValidationSizeLimit* - by default, Dataverse checks fixity (assuring the file contents match the recorded checksum) as part of publication. This setting specifies a maximum file size, above which validation will not be done.
- *:FilePIDsEnabled* - false is recommended when datasets have many files. Related settings allow file PIDS to be enabled/disabled per collection and per dataset
- *:CustomZipDownloadServiceUrl* - allows use of a separate process/machine to handle zipping up multi-file downloads. Requires installation of the separate Zip Download app
- *:WebloaderUrl* - enables use of an installed DVWebloader (by specifying its web location) which is more efficient for uploading many files
- *:CategoryOrder* - Pre-sorts the file display by category, e.g. showing all "Documentation" files before "Data" files. Any user selected sorting by name, age, or size is done within these sections
- *:OrderByFolder* - pre-sorts files by their directory Label (folder), showing files with no path before others. Any user selected sorting by name, age, or size is done within these sections
- *:DisableSolrFacets* - disables facets, which are costly to generate, in search results (including the main collection page)
- *:DisableSolrFacetsForGuestUsers* - only disable facets for guests

- *:DisableSolrFacetsWithoutJsession* - disables facets for users who have disabled cookies (e.g. for bots)
- *:DisableUncheckedTypesFacet* - only disables the facet showing the number of collections, datasets, files matching the query (this facet is potentially less useful than others)
- *:StoreIngestedTabularFilesWithVarHeaders* - by default, Dataverse stores ingested files without headers and dynamically adds them back at download time. Once this setting is enabled, Dataverse will leave the headers in place (for newly ingested files), reducing the cost of downloads
- *dataverse.bagit.zip.max-file-size*, *dataverse.bagit.zip.max-data-size*, and *dataverse.bagit.zip.holey* - options to control the size and temporary storage requirements when generating archival Bags - see *BagIt Export*

3.24.4 Scaling Infrastructure

There is no well-defined cut-off in terms of files per dataset or number of datasets where the Dataverse software will fail. In general the speed of viewing and editing a large dataset will decrease as the volume of datasets and files increases. For a given installation, at some point, Dataverse will need more memory than is available, or will max out the CPU or other resources and performance may decline dramatically.

In such cases:

- Consider increasing the memory available to Dataverse (the Java heap size for the Payara instance)
- Consider a larger machine (more CPU resources)
- Verify that performance isn't being limited by Solr or Postgres
- Investigate performance tuning options for Payara, Solr, and Postgres
- Coordinate with others in the community - there is a lot of aggregate knowledge
- Consider contributing to software design changes - Dataverse scaling has improved dramatically over the past several years, but more can be done
- Watch for the new single page application (SPA) front-end for Dataverse. It includes features such as infinite scrolling through files with much faster initial page load times

CHAPTER

FOUR

AI GUIDE

Artificial Intelligence (AI) is a growing component of the Dataverse ecosystem.

Contents:

- *Tools*
  - *Ask Dataverse*
  - *Ask the Data*
  - *AutoSage*
  - *DataChat*
  - *TurboCurator*
- *Protocols*
  - *Model Context Protocol (MCP)*

4.1 Tools

4.1.1 Ask Dataverse

Ask Dataverse (ask.dataverse.org) is a place to ask questions about the Dataverse Project and the Dataverse software. It was created by Slava Tykhonov who announced it in December 2024 and presented it February 2025 (video, slides, notes).

4.1.2 Ask the Data

Ask the Data is an *external tool* that allows you ask natural language questions about the data contained in Dataverse tables (tabular data). See the README.md file at https://github.com/IQSS/askdataverse/tree/main/askthedata for the instructions on adding Ask the Data to your Dataverse installation.

4.1.3 AutoSage

AutoSage provides metadata suggestions for datasets. Learn more at https://github.com/siacus/autosage.

4.1.4 DataChat

DataChat is a multilingual open source natural language interface for Dataverse and other data platforms with an experimental Graph AI implementation for Croissant support. DataChat can literally talk back to you and explain what is inside of every single dataset, you can ask any question and it responds on the level of metadata described by Croissant standard. Learn more at https://github.com/gdcc/datachat.

4.1.5 TurboCurator

TurboCurator is an *external tool* that generates metadata improvements for title, description, and keywords. It relies on OpenAI's ChatGPT & ICPSR best practices. See the TurboCurator Dataverse Administrator page for more details on how it works and adding TurboCurator to your Dataverse installation.

4.2 Protocols

4.2.1 Model Context Protocol (MCP)

Model Context Protocol (MCP) is a standard for AI Agents to communicate with tools and services, announced in November 2024.

An MCP server for Dataverse has been deployed to mcp.dataverse.org, powered by the code at https://github.com/gdcc/mcp-dataverse. See the code's README for information on configuring MCP clients (e.g. Cursor, Visual Studio Code, Windsurf, Zed, etc.) to use mcp.dataverse.org or your own local installation (setup instructions are also provided).

CHAPTER

FIVE

API GUIDE

Contents:

5.1 Introduction

The Dataverse Software APIs allow users to accomplish many tasks such as…

- creating datasets
- uploading files
- publishing datasets
- and much, much more

… all without using the Dataverse installation's web interface.

APIs open the door for integrations between the Dataverse Software and other software. For a list, see the *Integrations* section of the Admin Guide.

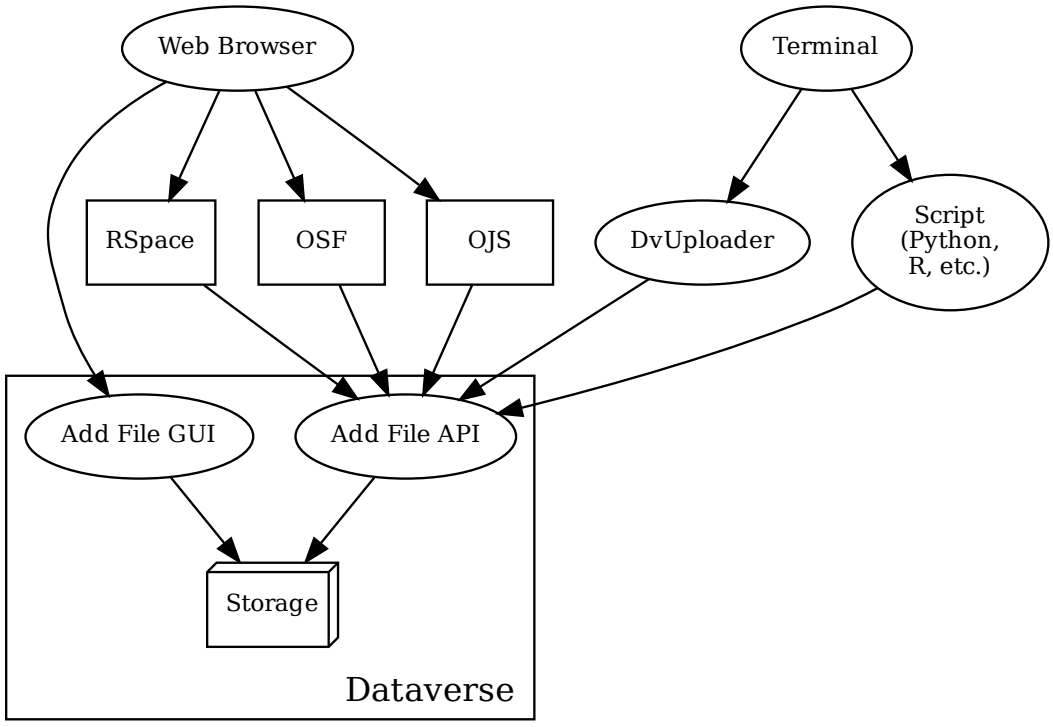
Contents:

- *What is an API?*
- *Types of Dataverse Software API Users*
  - *API Users Within a Single Dataverse Installation*
    - *Users of Integrations and Apps*
    - *Power Users*
    - *Support Teams and Superusers*
    - *Sysadmins*
    - *In House Developers*
  - *API Users Across the Dataverse Project*
    - *Developers of Integrations, External Tools, and Apps*
    - *Developers of Dataverse Software API Client Libraries*
    - *Developers of The Dataverse Software Itself*
- *How This Guide is Organized*

  - *Getting Started*
  - *API Tokens and Authentication*
  - *Lists of Dataverse APIs*
  - *Client Libraries*
  - *Examples*
  - *Frequently Asked Questions*
- *Getting Help*

5.1.1 What is an API?

API stands for "Application Programming Interface" and an example is the Dataverse Software's "file upload" API. In the diagram below, we can see that while users can click a button within Dataverse installation's web interface to upload a file, there are many other ways to get files into a Dataverse installation, all using an API that allows for uploading of files.



The components above that use the "file" upload API are:

- DvUploader is terminal-based application for uploading files that is described in the *Dataset + File Management* section of the User Guide.

- OJS, OSF, and RSpace are all web applications that can integrate with a Dataverse installation and are described in "Getting Data In" in the *Integrations* section of the Admin Guide.
- The script in the diagram can be as simple as a single line of code that is run in a terminal. You can copy and paste "one-liners" like this from the guide. See the *Getting Started with APIs* section for examples using a tool called "curl".

The diagram above shows only a few examples of software using a specific API but many more APIs are available.

5.1.2 Types of Dataverse Software API Users

This guide is intended to serve multiple audiences but pointers various sections of the guide are provided below based on the type of API user you are.

API Users Within a Single Dataverse Installation

Each Dataverse installation will have its own groups of people interested in APIs.

Users of Integrations and Apps

Integrations and apps can take many forms but two examples are:

- Using Open Science Framework (OSF), a web application, to deposit and publish data into a Dataverse installation.
- Using DVUploader, a terminal-based desktop application, to upload files into a Dataverse installation.

In both examples, users need to obtain an API Token to authenticate with a Dataverse installation.

A good starting point is "API Tokens" in the *Account Creation + Management* section of the User Guide. DvUploader is documented in the *Dataset + File Management* section of the User Guide. The integrations that are enabled depend on your Dataverse installation. You can find a list in the *Integrations* section of the Admin Guide.

Power Users

Power users may be researchers or curators who are comfortable with automating parts of their workflow by writing Python code or similar.

The recommended starting point for power users is the *Getting Started with APIs* section.

Support Teams and Superusers

Support teams that answer questions about their Dataverse installation should familiarize themselves with the *Getting Started with APIs* section to get a sense of common tasks that researchers and curators might be trying to accomplish by using Dataverse Software APIs.

Superusers of a Dataverse installation have access a superuser dashboard described in the *Dashboard* section of the Admin Guide but some operations can only be done via API.

A good starting point for both groups is the *Getting Started with APIs* section of this guide followed by the *Troubleshooting* section of the Admin Guide.

Sysadmins

Sysadmins often write scripts to automate tasks and Dataverse Software APIs make this possible. Sysadmins have control over the server that the Dataverse installation is running on and may be called upon to execute API commands that are limited to "localhost" (the server itself) for security reasons.

A good starting point for sysadmins is "Blocking API Endpoints" in the *Configuration* section of the Installation Guide, followed by the *Getting Started with APIs* section of this guide, followed by the *Troubleshooting* section of the Admin Guide.

In House Developers

Some organizations that run a Dataverse installation employ developers who are tasked with using the Dataverse installation's APIs to accomplish specific tasks such as building custom integrations with in house systems or creating reports specific to the organization's needs.

A good starting point for in house developers is the *Getting Started with APIs* section.

API Users Across the Dataverse Project

The Dataverse Project loves contributors! Depending on your interests and skills, you might fall into one or more of the groups below.

Developers of Integrations, External Tools, and Apps

One of the primary purposes for Dataverse Software APIs in the first place is to enable integrations with third party software. Integrations are listed in the following places:

- The *Integrations* section of the Admin Guide.
- The *Building External Tools* section this guide.
- The *Apps* section of this guide.

Good starting points are the three sections above to get a sense of third-party software that already integrates with the Dataverse Software, followed by the *Getting Started with APIs* section.

Developers of Dataverse Software API Client Libraries

A client library helps developers using a specific programming language such as Python, Javascript, R, or Java interact with Dataverse Software APIs in a manner that is idiomatic for their language. For example, a Python programmer may want to

A good starting point is the *Client Libraries* section, followed by the *Getting Started with APIs* section.

Developers of The Dataverse Software Itself

Developers working on the Dataverse Software itself use the APIs when adding features, fixing bugs, and testing those features and bug fixes.

A good starting point is the *Testing* section of the Developer Guide.

5.1.3 How This Guide is Organized

Getting Started

See *Getting Started with APIs*

API Tokens and Authentication

See *API Tokens and Authentication*.

Lists of Dataverse APIs

- *Search API*: For searching dataverse collections, datasets, and files.
- *Data Access API*: For downloading and subsetting data.
- *Native API*: For performing most tasks that are possible in the GUI. See *Getting Started with APIs* for the most common commands which operate on endpoints with names like:
  - Dataverses
  - Datasets
  - Files
  - etc.
- *Dataset Semantic Metadata API*: For creating, reading, editing, and deleting dataset metadata using JSON-LD.
- *Dataset Migration API*: For migrating datasets from other repositories while retaining the original persistent identifiers and publication date.
- *Direct DataFile Upload/Replace API*: For the transfer of larger files/larger numbers of files directly to an S3 bucket managed by Dataverse.
- *Globus Transfer API*: For the Globus transfer of larger files/larger numbers of files directly via Globus endpoints managed by Dataverse or referencing files in remote endpoints.
- *Metrics API*: For query statistics about usage of a Dataverse installation.
- *SWORD API*: For depositing data using a standards-based approach rather than the *Native API*.

Please note that some APIs are only documented in other guides that are more suited to their audience:

- Admin Guide
  - *External Tools*
  - *Metadata Customization*
  - *Metadata Export*
  - *Make Data Count*

  - *Solr Search Index*
- Installation Guide
  - *Configuration*
  - *Big Data Support*
- Developer Guide
  - *Auxiliary File Support*
  - *Dataset Migration API*
  - *Dataset Semantic Metadata API*
  - *Direct DataFile Upload/Replace API*
  - *Workflows*

Client Libraries

See *Client Libraries* for how to use Dataverse Software APIs from Python, R, Java, and Javascript.

Examples

Apps links to example open source code you can study. *Getting Started with APIs* also has many examples.

Frequently Asked Questions

See *Frequently Asked Questions*.

5.1.4 Getting Help

Dataverse Software API questions are on topic in all the usual places:

- The dataverse-community Google Group: https://groups.google.com/forum/#!forum/dataverse-community
- The Dataverse Project community calls: https://dataverse.org/community-calls
- The Dataverse Project chat room: https://chat.dataverse.org
- The Dataverse Project ticketing system: support@dataverse.org

After your question has been answered, you are welcome to help improve the *Frequently Asked Questions* section of this guide.

5.2 Getting Started with APIs

If you are a researcher or curator who wants to automate parts of your workflow, this section should help you get started. The *Introduction* section lists resources for other groups who may be interested in Dataverse Software APIs such as developers of integrations and support teams.

Contents:

- *Servers You Can Test With*
- *Getting an API Token*
- *curl Examples and Environment Variables*
- *Depositing Data*
  - *Creating a Dataverse Collection*
  - *Creating a Dataset*
  - *Uploading Files*
  - *Publishing a Dataverse Collection*
  - *Publishing a Dataset*
- *Finding and Downloading Data*
  - *Finding Datasets*
  - *Finding Recently Published Dataverse Collections, Datasets, and Files*
  - *Downloading Files*
  - *Downloading Metadata*
  - *Listing the Contents of a Dataverse Collection*
- *Managing Permissions*
  - *Granting Permission*
  - *Revoking Permission*
  - *Listing Permissions (Role Assignments)*
- *Getting the OpenAPI Document*
- *Beyond "Getting Started" Tasks*
- *Getting Help*

5.2.1 Servers You Can Test With

Rather than using a production Dataverse installation, API users are welcome to use https://demo.dataverse.org for testing. You can email support@dataverse.org if you have any trouble with this server.

If you would rather have full control over your own test server, deployments to AWS, Docker, and more are covered in the *Developer Guide* and the *Installation Guide*.

5.2.2 Getting an API Token

Many Dataverse Software APIs require an API token.

Once you have identified a server to test with, create an account, click on your name, and get your API token. For more details, see the *API Tokens and Authentication* section.

5.2.3 curl Examples and Environment Variables

The examples in this guide use curl for the following reasons:

- curl commands are succinct.
- curl commands can be copied and pasted into a terminal.
- This guide is programming language agnostic. It doesn't prefer any particular programming language.

You'll find curl examples that look like this:

```
export SERVER_URL=https://demo.dataverse.org
export QUERY=data

curl $SERVER_URL/api/search?q=$QUERY
```

What's going on above is the declaration of "environment variables" that are substituted into a curl command. You should run the "export" commands but change the value for the server URL or the query (or whatever options the command supports). Then you should be able to copy and paste the curl command and it should "just work", substituting the variables like this:

```
curl https://demo.dataverse.org/api/search?q=data
```

If you ever want to check an environment variable, you can "echo" it like this:

```
echo $SERVER_URL
```

With curl version 7.56.0 and higher, it is recommended to use –form-string with outer quote rather than -F flag without outer quote.

For example, curl command parameter below might cause error such as `warning: garbage at end of field specification: ,"categories":["Data"]}`.

```
-F jsonData={\"description\":\"My description.\",\"categories\":[\"Data\"]}
```

Instead, use –form-string with outer quote. See https://github.com/curl/curl/issues/2022

```
--form-string 'jsonData={"description":"My description.","categories":["Data"]}'
```

If you don't like curl, don't have curl, or want to use a different programming language, you are encouraged to check out the Python, Javascript, R, and Java options in the *Client Libraries* section.

5.2.4 Depositing Data

Creating a Dataverse Collection

See *Create a Dataverse Collection*.

Creating a Dataset

See *Create a Dataset in a Dataverse Collection*.

Uploading Files

See *Add a File to a Dataset*. In addition, when a Dataverse installation is configured to use S3 storage with direct upload enabled, there is API support to send a file directly to S3. This facilitates an efficient method to upload big files, but is more complex. The procedure is described in the *Direct DataFile Upload/Replace API* section of the Developer Guide.

Publishing a Dataverse Collection

See *Publish a Dataverse Collection*.

Publishing a Dataset

See *Update Dataset Terms of Access*.

5.2.5 Finding and Downloading Data

Finding Datasets

A quick example search for the word "data" is https://demo.dataverse.org/api/search?q=data

See the *Search API* section for details.

Finding Recently Published Dataverse Collections, Datasets, and Files

See *Date Range Search Example*.

It's also possible to find recently published datasets via OAI-PMH.

Downloading Files

The *Data Access API* section explains how to download files.

To download all the files in a dataset, see *Downloading All Files in a Dataset*.

In order to download individual files, you must know their database IDs which you can get from the `dataverse_json` metadata at the dataset level. See *Export Metadata of a Dataset in Various Formats*.

Downloading Metadata

Dataset metadata is available in a variety of formats listed at *Supported Metadata Export Formats*.

See *Export Metadata of a Dataset in Various Formats*.

Listing the Contents of a Dataverse Collection

See *Show Contents of a Dataverse Collection*.

5.2.6 Managing Permissions

Granting Permission

See *Assign a New Role on a Dataverse Collection*.

Revoking Permission

See *Delete Role Assignment from a Dataverse Collection*.

Listing Permissions (Role Assignments)

See *List Role Assignments in a Dataverse Collection*.

5.2.7 Getting the OpenAPI Document

You can access our OpenAPI document using the `/openapi` endpoint. The default format is YAML if no parameter is provided, but you can also obtain the JSON version by either passing `format=json` as a query parameter or by sending `Accept:application/json` (case-sensitive) as a header.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org
export FORMAT=json

curl "$SERVER_URL/openapi?format=$FORMAT"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/openapi?format=json"
```

If you have feedback about the our OpenAPI document, please contact us through the usual channels under *Getting Help*.

You can find more information about validating the document under *OpenAPI* in the Developer Guide.

5.2.8 Beyond "Getting Started" Tasks

In addition to the tasks listed above, your Dataverse installation supports many other operations via API.

See *Lists of Dataverse APIs* and *Types of Dataverse Software API Users* to get oriented.

If you're looking for some inspiration for how you can use the Dataverse Software APIs, there are open source projects that integrate with the Dataverse Software listed in the *Apps* section.

5.2.9 Getting Help

See *Getting Help*.

5.3 API Tokens and Authentication

An API token is similar to a password and allows you to authenticate to Dataverse Software APIs to perform actions as you. Many Dataverse Software APIs require the use of an API token.

Contents:

- *How to Get an API Token*
- *How Your API Token Is Like a Password*
- *Passing Your API Token as an HTTP Header (Preferred) or a Query Parameter*
- *Resetting Your API Token*
- *Using Preview URL Tokens as API Token*
- *Bearer Tokens*
- *Signed URLs*

5.3.1 How to Get an API Token

Your API token is unique to the server you are using. You cannot take your API token from one server and use it on another server.

Instructions for getting a token are described in the *Account Creation + Management* section of the User Guide.

5.3.2 How Your API Token Is Like a Password

Anyone who has your API Token can add and delete data as you so you should treat it with the same care as a password.

5.3.3 Passing Your API Token as an HTTP Header (Preferred) or a Query Parameter

Note: The SWORD API uses a different way of passing the API token. Please see *Authentication for SWORD* for details.

See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

There are two ways to pass your API token to Dataverse Software APIs. The preferred method is to send the token in the `X-Dataverse-key` HTTP header, as in the following curl example.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ALIAS=root

curl -H X-Dataverse-key:$API_TOKEN $SERVER_URL/api/dataverses/$ALIAS/contents
```

Here's how it looks without the environment variables:

```
curl -H X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx https://demo.dataverse.org/
↪api/dataverses/root/contents
```

The second way to pass your API token is via a query parameter called `key` in the URL like below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ALIAS=root

curl $SERVER_URL/api/dataverses/$ALIAS/contents?key=$API_TOKEN
```

Here's how it looks without the environment variables:

```
curl https://demo.dataverse.org/api/dataverses/root/contents?key=xxxxxxxx-xxxx-xxxx-xxxx-
↪xxxxxxxxxxxx
```

Use of the `X-Dataverse-key` HTTP header form is preferred to passing `key` in the URL because query parameters like `key` appear in URLs and might accidentally get shared, exposing your API token. (Again it's like a password.) Additionally, URLs are often logged on servers while it's less common to log HTTP headers.

5.3.4 Resetting Your API Token

You can reset your API Token from your account page in your Dataverse installation as described in the *Account Creation + Management* section of the User Guide.

5.3.5 Using Preview URL Tokens as API Token

If you want to access a dataset's contents shared by a preview URL (see *Preview URL to Review Unpublished Dataset*, *Using a Preview URL Token*, *Get Dataset By Preview URL Token* and *Create a Preview URL for a Dataset*) using the API, you can use the token in the preview URL as API_TOKEN.

```
SERVER_URL=https://demo.dataverse.org
PREVIEW_URL_TOKEN=a56444bc-7697-4711-8964-e0577f055fd2

# this gets you the file listing with the corresponding file IDs, file names and other␣
```

(continues on next page)

(continued from previous page)

```
↪file metadata
curl "$SERVER_URL/api/datasets/privateUrlDatasetVersion/$PREVIEW_URL_TOKEN"

# this gets you a specific file
FILE_ID=1111111
FILENAME=example.txt
curl -H "X-Dataverse-key:$PREVIEW_URL_TOKEN" -o "$FILENAME" "$SERVER_URL/api/access/
↪datafile/$FILE_ID"
```

5.3.6 Bearer Tokens

Bearer tokens are defined in RFC 6750 and can be used as an alternative to API tokens if your installation has been set up to use them (see *Bearer Token Authentication* in the Installation Guide).

To test if bearer tokens are working, you can try something like the following (using the *User Information* API endpoint), substituting in parameters for your installation and user.

```
export TOKEN=`curl -s -X POST --location "http://keycloak.mydomain.com:8090/realms/test/
↪protocol/openid-connect/token" -H "Content-Type: application/x-www-form-urlencoded" -d
↪"username=user&password=user&grant_type=password&client_id=test&client_
↪secret=94XHrfNRwXsjqTqApRrwWmhDLDHpIYV8" | jq '.access_token' -r | tr -d "\n"`

curl -H "Authorization: Bearer $TOKEN" http://localhost:8080/api/users/:me
```

To register a new user who has authenticated via an OIDC provider, the following endpoint should be used:

```
curl  -H "Authorization: Bearer $TOKEN" -X POST http://localhost:8080/api/users/register␣
↪--data '{"termsAccepted":true}'
```

By default, the Bearer token is expected to include the following claims that will be used to create the user account:

- `username`
- `firstName`
- `lastName`
- `emailAddress`

The one parameter required by default is `termsAccepted` which must be set to true, indicating that the user has seen and accepted the Terms of Use of the installation.

If the feature flag `api-bearer-auth-handle-tos-acceptance-in-idp` is enabled (along with the `api-bearer-auth` feature flag), Dataverse assumes that the Terms of Service acceptance was handled by the identity provider, e.g. in the OIDC `consent` dialog, and the `termsAccepted` parameter is not needed.

There is another flag called `api-bearer-auth-provide-missing-claims` that can be enabled (along with the `api-bearer-auth` feature flag) to allow sending missing user claims in the registration JSON. This is useful when the identity provider does not supply the necessary claims listed above. If properties are provided in the JSON, but corresponding claims already exist in the identity provider, an error will be thrown, outlining the conflicting properties. Note that supplying missing claims is configured via a separate feature flag because using it may introduce user impersonation issues, for example if the identity provider does not provide an email field and the user submits an email address they do not own.

In all cases, the submitted JSON can optionally include the fields `position` or `affiliation`, which will be added to the user's Dataverse account. These fields are optional, and if not provided, they will be persisted as empty in Dataverse.

5.3.7 Signed URLs

See *Signed URLs*.

5.4 Search API

Contents:

- *Parameters*
- *Basic Search Example*
- *Advanced Search Examples*
  - *Narrowed to Collection, Show Relevance and Facets*
  - *Retrieve Released Versions Only*
  - *Include Metadata Blocks and/or Metadata Fields*
  - *Include Specific Fields Only*
- *Date Range Search Example*
- *Iteration*
- *Search Services*
  - *Discovering Available Search Services*
  - *Using Different Search Services*
  - *Developing Search Services*

The Search API supports the same searching, sorting, and faceting operations as the Dataverse Software's web interface.

To search unpublished content, you must pass in an API token as described in the *API Tokens and Authentication* section.

The parameters and JSON response are partly inspired by the GitHub Search API.

Note

The search API can be used from scripts running in web browsers, as it allows cross-origin resource sharing (CORS).

Please note that in Dataverse Software 4.3 and older the "citation" field wrapped the persistent ID URL in an <a> tag but this has been changed to plaintext. If you want the old value with HTML in it, a new field called "citationHtml" can be used.

5.4.1 Parameters

Nam	Type	Description
q	string	The search term or terms. Using "title:data" will search only the "title" field. "*" can be used as a wildcard either alone or adjacent to a term (i.e. "bird*"). For example, https://demo.dataverse.org/api/search?q=title:data . For a list of fields to search, please see https://github.com/IQSS/dataverse/issues/2558 (for now).
type	string	Can be either "dataverse", "dataset", or "file". Multiple "type" parameters can be used to include multiple types (i.e. `type=dataset&type=file`). If omitted, all types will be returned. For example, https://demo.dataverse.org/api/search?q=*&type=dataset
subtree	string	The identifier of the Dataverse collection to which the search should be narrowed. The subtree of this Dataverse collection and all its children will be searched. Multiple "subtree" parameters can be used to include multiple Dataverse collections. For example, https://demo.dataverse.org/api/search?q=data&subtree=birds&subtree=cats .
sort	string	The sort field. Supported values include "name", "date", and "score". Sorting by "score" orders by **relevance** and is the default if this parameter is omitted.
order	string	The order in which to sort. Can either be "asc" or "desc". For example, https://demo.dataverse.org/api/search?q=data&sort=name&order=asc
per_p	int	The number of results to return per request. The default is 10. The max is 1000. See *iteration example*.
start	int	A cursor for paging through search results. See *iteration example*.
show	boole	Whether or not to show details of which fields were matched by the query. False by default. See *advanced search example*.
show	boole	Whether or not to show facets that can be operated on by the "fq" parameter. False by default. See *advanced search example*.
fq	string	A filter query on the search term. Multiple "fq" parameters can be used. See *advanced search example*.
show	boole	Whether or not to show the database IDs of the search results (for developer use).
show	boole	Whether or not to show API URLs for the search results
query	boole	Whether to query entities for extra metadata (slower). Default is true.
metadata_	string	Includes the requested fields for each dataset in the response. Multiple "metadata_fields" parameters can be used to include several fields. The value must be in the form "{metadata_block_name}:{field_name}" to include a specific field from a metadata block (see *example*) or "{metadata_field_set_name}:*" to include all the fields for a metadata block (see *example*). "{field_name}" cannot be a subfield of a compound field. If "{field_name}" is a compound field, all subfields are included.
geo_p	string	Latitude and longitude in the form `geo_point=42.3,-71.1`. You must supply `geo_radius` as well. See also *Geospatial Search*.
geo_r	string	Radial distance in kilometers from `geo_point` (which must be supplied as well) such as `geo_radius=1.5`.
show	boole	Whether or not to include total_count_per_object_type for types: Dataverse, Dataset, and Files.
show	boole	Whether or not to include a list of parent and linked collections for each dataset search result.
searc	string	The name of the search service to use for this query. If omitted, the default search service will be used. For available search services, see *Discovering Available Search Services*.

5.4.2 Basic Search Example

https://demo.dataverse.org/api/search?q=trees


```
{
    "status":"OK",
    "data":{
        "q":"trees",
        "total_count":5,
        "start":0,
        "spelling_alternatives":{
            "trees":"[tree]"
        },
        "items":[
            {
                "name":"Trees",
                "type":"dataverse",
                "url":"https://demo.dataverse.org/dataverse/trees",
                "image_url":"https://demo.dataverse.org/api/access/dvCardImage/1",
                "identifier":"trees",
                "description":"A tree dataverse with some birds",
                "published_at":"2016-05-10T12:53:38Z",
                "publicationStatuses": [
                    "Published"
                ],
                "affiliation": "Dataverse.org",
                "parentDataverseName": "Root",
                "parentDataverseIdentifier": "root"
            },
            {
                "name":"Chestnut Trees",
                "type":"dataverse",
                "url":"https://demo.dataverse.org/dataverse/chestnuttrees",
                "image_url":"https://demo.dataverse.org/api/access/dvCardImage/2",
                "identifier":"chestnuttrees",
                "description":"A dataverse with chestnut trees and an oriole",
                "published_at":"2016-05-10T12:52:38Z",
                "publicationStatuses": [
                    "Published"
                ],
                "affiliation": "Dataverse.org",
                "parentDataverseName": "Root",
                "parentDataverseIdentifier": "root"
            },
            {
                "name":"trees.png",
                "type":"file",
                "url":"https://demo.dataverse.org/api/access/datafile/12",
                "image_url":"https://demo.dataverse.org/api/access/datafile/12?
↪imageThumb=true",
                "file_id":"12",
                "description":"",
                "published_at":"2016-05-10T12:53:39Z",
```


(continues on next page)

(continued from previous page)


```
                "file_type":"PNG Image",
                "file_content_type":"image/png",
                "size_in_bytes":8361,
                "md5":"0386269a5acb2c57b4eade587ff4db64",
                "file_persistent_id": "doi:10.5072/FK2/XTT5BV/PCCHV7",
                "dataset_name": "Dataset One",
                "dataset_id": "32",
                "dataset_persistent_id": "doi:10.5072/FK2/XTT5BV",
                "dataset_citation":"Spruce, Sabrina, 2016, \"Spruce Goose\", http://dx.
↪doi.org/10.5072/FK2/XTT5BV, Root Dataverse, V1",
                "publicationStatuses": [
                    "Published"
                ],
                "releaseOrCreateDate": "2016-05-10T12:53:39Z"
            },
            {
                "name":"Birds",
                "type":"dataverse",
                "url":"https://demo.dataverse.org/dataverse/birds",
                "image_url":"https://demo.dataverse.org/api/access/dvCardImage/3",
                "identifier":"birds",
                "description":"A bird Dataverse collection with some trees",
                "published_at":"2016-05-10T12:57:27Z",
                "publicationStatuses": [
                    "Published"
                ],
                "affiliation": "Dataverse.org",
                "parentDataverseName": "Root",
                "parentDataverseIdentifier": "root"
            },
            {
                "name":"Darwin's Finches",
                "type":"dataset",
                "url":"https://doi.org/10.70122/FK2/MB5VGR",
                "global_id":"doi:10.70122/FK2/MB5VGR",
                "description":"Darwin's finches (also known as the GalÃ¡pagos finches)␣
↪are a group of about fifteen species of passerine birds.",
                "published_at":"2019-12-11T15:26:10Z",
                "publisher":"dvbe69f5e1",
                "citationHtml":"Finch, Fiona; Spruce, Sabrina; Poe, Edgar Allen;␣
↪Mulligan, Hercules, 2019, \"Darwin's Finches\", <a href=\"https://doi.org/10.70122/FK2/
↪MB5VGR\" target=\"_blank\">https://doi.org/10.70122/FK2/MB5VGR</a>, Root, V3",
                "identifier_of_dataverse":"dvbe69f5e1",
                "name_of_dataverse":"dvbe69f5e1",
                "citation":"Finch, Fiona; Spruce, Sabrina; Poe, Edgar Allen; Mulligan,␣
↪Hercules, 2019, \"Darwin's Finches\", https://doi.org/10.70122/FK2/MB5VGR, Root, V3",
                "publicationStatuses": [
                    "Published"
                ],
                "storageIdentifier":"file://10.70122/FK2/MB5VGR",
                "subjects":[
                   "Astronomy and Astrophysics",
```


(continues on next page)

(continued from previous page)


```
                    "Other"
                ],
                "fileCount":3,
                "versionId":1260,
                "versionState":"RELEASED",
                "majorVersion":3,
                "minorVersion":0,
                "createdAt":"2019-09-20T18:08:29Z",
                "updatedAt":"2019-12-11T15:26:10Z",
                "contacts":[
                    {
                        "name":"Finch, Fiona",
                        "affiliation":""
                    }
                ],
                "producers":[
                    "Allen, Irwin",
                    "Spielberg, Stephen"
                ],
                "authors":[
                    "Finch, Fiona",
                    "Spruce, Sabrina",
                    "Poe, Edgar Allen",
                    "Mulligan, Hercules"
                ]
            }
        ],
        "count_in_response":5
    }
}
```


5.4.3 Advanced Search Examples

Narrowed to Collection, Show Relevance and Facets

https://demo.dataverse.org/api/search?q=finch&show_relevance=true&show_facets=true&fq=publicationDate:2016&subtree=birds

In this example, `show_relevance=true` matches per field are shown. Available facets are shown with `show_facets=true` and of the facets is being used with `fq=publicationDate:2016`. The search is being narrowed to the Dataverse collection with the identifier "birds" with the parameter `subtree=birds`.


```
{
    "status":"OK",
    "data":{
        "q":"finch",
        "total_count":2,
        "start":0,
        "spelling_alternatives":{
        },
        "items":[
```


(continues on next page)

(continued from previous page)


```
            {
                "name":"Finches",
                "type":"dataverse",
                "url":"https://demo.dataverse.org/dataverse/finches",
                "image_url":"https://demo.dataverse.org/api/access/dvCardImage/2",
                "identifier":"finches",
                "description":"A Dataverse collection with finches",
                "published_at":"2016-05-10T12:57:38Z",
                "matches":[
                    {
                        "description":{
                            "snippets":[
                                "A Dataverse collection with <span class=\"search-term-
↪match\">finches</span>"
                            ]
                        }
                    },
                    {
                        "name":{
                            "snippets":[
                                "<span class=\"search-term-match\">Finches</span>"
                            ]
                        }
                    }
                ],
                "score": 3.8500118255615234
            },
            {
                "name":"Darwin's Finches",
                "type":"dataset",
                "url":"http://dx.doi.org/10.5072/FK2/G2VPE7",
                "image_url":"https://demo.dataverse.org/api/access/dsCardImage/2",
                "global_id":"doi:10.5072/FK2/G2VPE7",
                "description": "Darwin's finches (also known as the Galápagos finches)␣
↪are a group of about fifteen species of passerine birds.",
                "published_at":"2016-05-10T12:57:45Z",
                "citationHtml":"Finch, Fiona, 2016, \"Darwin's Finches\", <a href=\
↪"http://dx.doi.org/10.5072/FK2/G2VPE7\" target=\"_blank\">http://dx.doi.org/10.5072/
↪FK2/G2VPE7</a>, Root Dataverse, V1",
                "citation":"Finch, Fiona, 2016, \"Darwin's Finches\", http://dx.doi.org/
↪10.5072/FK2/G2VPE7, Root Dataverse, V1",
                "publicationStatuses": [
                    "Published"
                ],
                "matches":[
                    {
                        "authorName":{
                            "snippets":[
                                "<span class=\"search-term-match\">Finch</span>, Fiona"
                            ]
                        }
                    },
```


(continues on next page)

(continued from previous page)


```
                    {
                        "dsDescriptionValue":{
                            "snippets":[
                                "Darwin's <span class=\"search-term-match\">finches</
↪span> (also known as the Galápagos <span class=\"search-term-match\">finches</span>)␣
↪are a group of about fifteen species"
                            ]
                        }
                    },
                    {
                        "title":{
                            "snippets":[
                                "Darwin's <span class=\"search-term-match\">Finches</
↪span>"
                            ]
                        }
                    }
                ],
                "score": 1.5033848285675049,
                "authors":[
                    "Finch, Fiona"
                ]
            }
        ],
        "facets":[
            {
                "subject_ss":{
                    "friendly":"Subject",
                    "labels":[
                        {
                            "Medicine, Health and Life Sciences":2
                        }
                    ]
                },
                "authorName_ss": {
                    "friendly":"Author Name",
                    "labels": [
                        {
                            "Finch, Fiona":1
                        }
                    ]
                },
                "publicationDate":{
                    "friendly":"Publication Date",
                    "labels":[
                        {
                            "2016":2
                        }
                    ]
                }
            }
        ],
```


(continues on next page)

(continued from previous page)

```
        "count_in_response":2
    }
}
```

Retrieve Released Versions Only

https://demo.dataverse.org/api/search?q=finch&fq=publicationStatus:Published&type=dataset

The above example `fq=publicationStatus:Published` retrieves only "RELEASED" versions of datasets. The same could be done to retrieve "DRAFT" versions, `fq=publicationStatus:Draft`


```
{
    "status": "OK",
    "data": {
        "q": "finch",
        "total_count": 2,
        "start": 0,
        "spelling_alternatives": {},
        "items": [
            {
                "name": "Darwin's Finches",
                "type": "dataset",
                "url": "https://doi.org/10.70122/FK2/GUAS41",
                "global_id": "doi:10.70122/FK2/GUAS41",
                "description": "Darwin's finches (also known as the Galápagos finches)␣
↪are a group of about fifteen species of passerine birds.",
                "published_at": "2019-12-24T08:05:02Z",
                "publisher": "mdmizanur rahman Dataverse collection",
                "citationHtml": "Finch, Fiona, 2019, \"Darwin's Finches\", <a href=\
↪"https://doi.org/10.70122/FK2/GUAS41\" target=\"_blank\">https://doi.org/10.70122/FK2/
↪GUAS41</a>, Demo Dataverse, V1",
                "identifier_of_dataverse": "rahman",
                "name_of_dataverse": "mdmizanur rahman Dataverse collection",
                "citation": "Finch, Fiona, 2019, \"Darwin's Finches\", https://doi.org/
↪10.70122/FK2/GUAS41, Demo Dataverse, V1",
                "publicationStatuses": [
                    "Published"
                ],
                "storageIdentifier": "file://10.70122/FK2/GUAS41",
                "subjects": [
                    "Medicine, Health and Life Sciences"
                ],
                "fileCount":6,
                "versionId": 53001,
                "versionState": "RELEASED",
                "majorVersion": 1,
                "minorVersion": 0,
                "createdAt": "2019-12-05T09:18:30Z",
                "updatedAt": "2019-12-24T08:38:00Z",
                "contacts": [
                    {
```


(continues on next page)

(continued from previous page)


```
                        "name": "Finch, Fiona",
                        "affiliation": ""
                    }
                ],
                "authors": [
                    "Finch, Fiona"
                ]
            },
            {
                "name": "Darwin's Finches",
                "type": "dataset",
                "url": "https://doi.org/10.70122/FK2/7ZXYRH",
                "global_id": "doi:10.70122/FK2/7ZXYRH",
                "description": "Darwin's finches (also known as the Galápagos finches) are a group of about fifteen species of passerine birds.",
                "published_at": "2020-01-22T21:47:34Z",
                "publisher": "Demo Dataverse",
                "citationHtml": "Finch, Fiona, 2020, \"Darwin's Finches\", <a href=\"https://doi.org/10.70122/FK2/7ZXYRH\" target=\"_blank\">https://doi.org/10.70122/FK2/7ZXYRH</a>, Demo Dataverse, V1",
                "identifier_of_dataverse": "demo",
                "name_of_dataverse": "Demo Dataverse",
                "citation": "Finch, Fiona, 2020, \"Darwin's Finches\", https://doi.org/10.70122/FK2/7ZXYRH, Demo Dataverse, V1",
                "publicationStatuses": [
                    "Published"
                ],
                "storageIdentifier": "file://10.70122/FK2/7ZXYRH",
                "subjects": [
                    "Medicine, Health and Life Sciences"
                ],
                "fileCount":9,
                "versionId": 53444,
                "versionState": "RELEASED",
                "majorVersion": 1,
                "minorVersion": 0,
                "createdAt": "2020-01-22T21:23:43Z",
                "updatedAt": "2020-01-22T21:47:34Z",
                "contacts": [
                    {
                        "name": "Finch, Fiona",
                        "affiliation": ""
                    }
                ],
                "authors": [
                    "Finch, Fiona"
                ]
            }
        ],
        "count_in_response": 2
    }
}
```

Include Metadata Blocks and/or Metadata Fields

https://demo.dataverse.org/api/search?q=*&type=dataset&metadata_fields=citation:*

The above example `metadata_fields=citation:*` returns under "metadataBlocks" all fields from the "citation" metadata block.


```
{
    "status": "OK",
    "data": {
        "q": "*",
        "total_count": 4,
        "start": 0,
        "spelling_alternatives": {},
        "items": [
            {
                "name": "JDD avec GeoJson 2021-07-13T10:23:46.409Z",
                "type": "dataset",
                "url": "https://doi.org/10.5072/FK2/GIWCKB",
                "global_id": "doi:10.5072/FK2/GIWCKB",
                "description": "Démo sprint 5. Cette couche représente l'emprise des␣
↪cimetières sur le territoire des Métropole. Ces périmètres d'emprise des cimetières␣
↪sont issus du recensement des informations des PLU/POS de chaque commune de la␣
↪métropole, des données du cadastre DGFiP et d'un inventaire terrain du Service␣
↪Planification et Études Urbaines de Métropole",
                "publisher": "Sample Data",
                "citationHtml": "Rennes M&eacute;tropole, 2021, \"JDD avec GeoJson 2021-
↪07-13T10:23:46.409Z\", <a href=\"https://doi.org/10.5072/FK2/GIWCKB\" target=\"_blank\
↪">https://doi.org/10.5072/FK2/GIWCKB</a>, Root, DRAFT VERSION",
                "identifier_of_dataverse": "Sample_data",
                "name_of_dataverse": "Sample Data",
                "citation": "Métropole, 2021, \"JDD avec GeoJson 2021-07-13T10:23:46.
↪409Z\", https://doi.org/10.5072/FK2/GIWCKB, Root, DRAFT VERSION",
                "publicationStatuses": [
                    "Unpublished",
                    "Draft"
                ],
                "storageIdentifier": "file://10.5072/FK2/GIWCKB",
                "subjects": [
                    "Other"
                ],
                "fileCount": 0,
                "versionId": 9976,
                "versionState": "DRAFT",
                "createdAt": "2021-07-13T10:28:45Z",
                "updatedAt": "2021-07-13T10:28:45Z",
                "contacts": [
                    {
                        "name": "string",
                        "affiliation": "string"
                    }
                ],
                "metadataBlocks": {
                    "citation": {
```


(continues on next page)

(continued from previous page)

```
                        "displayName": "Citation Metadata",
                        "fields": [
                            {
                                "typeName": "dsDescription",
                                "multiple": true,
                                "typeClass": "compound",
                                "value": [
                                    {
                                        "dsDescriptionValue": {
                                            "typeName": "dsDescriptionValue",
                                            "multiple": false,
                                            "typeClass": "primitive",
                                            "value": "Démo sprint 5. Cette couche␣
↪représente l'emprise des cimetières sur le territoire des Métropole. Ces périmètres d
↪'emprise des cimetières sont issus du recensement des informations des PLU/POS de␣
↪chaque commune de la métropole, des données du cadastre DGFiP et d'un inventaire␣
↪terrain du Service Planification et Études Urbaines de Métropole"
                                        },
                                        "dsDescriptionDate": {
                                            "typeName": "dsDescriptionDate",
                                            "multiple": false,
                                            "typeClass": "primitive",
                                            "value": "2021-07-13"
                                        }
                                    }
                                ]
                            },
                            {
                                "typeName": "author",
                                "multiple": true,
                                "typeClass": "compound",
                                "value": [
                                    {
                                        "authorName": {
                                            "typeName": "authorName",
                                            "multiple": false,
                                            "typeClass": "primitive",
                                            "value": "Métropole"
                                        },
                                        "authorAffiliation": {
                                            "typeName": "authorAffiliation",
                                            "multiple": false,
                                            "typeClass": "primitive",
                                            "value": "string"
                                        }
                                    }
                                ]
                            },
                            {
                                "typeName": "datasetContact",
                                "multiple": true,
                                "typeClass": "compound",
```

(continues on next page)

(continued from previous page)


```
                            "value": [
                                {
                                    "datasetContactName": {
                                        "typeName": "datasetContactName",
                                        "multiple": false,
                                        "typeClass": "primitive",
                                        "value": "string"
                                    },
                                    "datasetContactAffiliation": {
                                        "typeName": "datasetContactAffiliation",
                                        "multiple": false,
                                        "typeClass": "primitive",
                                        "value": "string"
                                    },
                                    "datasetContactEmail": {
                                        "typeName": "datasetContactEmail",
                                        "multiple": false,
                                        "typeClass": "primitive",
                                        "value": "contact@Sample.fr"
                                    }
                                }
                            ]
                        },
                        {
                            "typeName": "subject",
                            "multiple": true,
                            "typeClass": "controlledVocabulary",
                            "value": [
                                "Other"
                            ]
                        },
                        {
                            "typeName": "title",
                            "multiple": false,
                            "typeClass": "primitive",
                            "value": "JDD avec GeoJson 2021-07-13T10:23:46.409Z"
                        }
                    ]
                }
            },
            "authors": [
                "Métropole"
            ]
        },
        {
            "name": "Raja Ampat Islands",
            "type": "dataset",
            "url": "https://doi.org/10.5072/FK2/ITNXGR",
            "global_id": "doi:10.5072/FK2/ITNXGR",
            "description": "Raja Ampat is located off the northwest tip of Bird's␣
↪Head Peninsula on the island of New Guinea, in Indonesia's West Papua province, Raja␣
↪Ampat, or the Four Kings, is an archipelago comprising over 1,500 small islands, cays,␣
```


(continues on next page)

(continued from previous page)


```
↪and shoals surrounding the four main islands of Misool, Salawati, Batanta, and Waigeo,␣
↪and the smaller island of Kofiau. The Raja Ampat archipelago straddles the Equator and␣
↪forms part of Coral Triangle which contains the richest marine biodiversity on earth.␣
↪Administratively, the archipelago is part of the province of West Papua (formerly␣
↪known as Irian Jaya). Most of the islands constitute the Raja Ampat Regency, which was␣
↪separated out from Sorong Regency in 2004. The regency encompasses around 70,000␣
↪square kilometres (27,000 sq mi) of land and sea, and has a population of about 50,000␣
↪(as of 2017). (Wikipedia: https://en.wikipedia.org/wiki/Raja_Ampat_Islands)",
                "published_at": "2020-07-30T09:23:34Z",
                "publisher": "Root",
                "citationHtml": "Admin, Dataverse, 2020, \"Raja Ampat Islands\", <a␣
↪href=\"https://doi.org/10.5072/FK2/ITNXGR\" target=\"_blank\">https://doi.org/10.5072/
↪FK2/ITNXGR</a>, Root, V1",
                "identifier_of_dataverse": "root",
                "name_of_dataverse": "Root",
                "citation": "Admin, Dataverse, 2020, \"Raja Ampat Islands\", https://doi.
↪org/10.5072/FK2/ITNXGR, Root, V1",
                "authors": [
                    "Admin, Dataverse"
                ]
            },
            {
                "name": "Sample Test",
                "type": "dataverse",
                "url": "https://68b2d8bb37c6/dataverse/Sample_test",
                "identifier": "Sample_test",
                "description": "Dataverse utilisé pour les tests unitaires de Sample",
                "published_at": "2021-03-16T08:11:54Z"
            },
            {
                "name": "Sample Media Test",
                "type": "dataverse",
                "url": "https://68b2d8bb37c6/dataverse/Sample_media_test",
                "identifier": "Sample_media_test",
                "description": "Dataverse de test contenant les médias de Sample, comme␣
↪les images des fournisseurs et des producteurs",
                "published_at": "2021-04-08T15:04:14Z"
            }
        ],
        "count_in_response": 4
    }
}
```

Include Specific Fields Only

https://demo.dataverse.org/api/search?q=*&type=dataset&metadata_fields=citation:dsDescription&metadata_fields=citation:author

The above example `metadata_fields=citation:dsDescription&metadata_fields=citation:author` returns under "metadataBlocks" only the compound fields "dsDescription" and "author" metadata fields from the "citation" metadata block.


```
{
    "status": "OK",
    "data": {
        "q": "*",
        "total_count": 4,
        "start": 0,
        "spelling_alternatives": {},
        "items": [
            {
                "name": "JDD avec GeoJson 2021-07-13T10:23:46.409Z",
                "type": "dataset",
                "url": "https://doi.org/10.5072/FK2/GIWCKB",
                "global_id": "doi:10.5072/FK2/GIWCKB",
                "description": "Démo sprint 5. Cette couche représente l'emprise des␣
↪cimetières sur le territoire des Métropole. Ces périmètres d'emprise des cimetières␣
↪sont issus du recensement des informations des PLU/POS de chaque commune de la␣
↪métropole, des données du cadastre DGFiP et d'un inventaire terrain du Service␣
↪Planification et Études Urbaines de Métropole",
                "publisher": "Sample Data",
                "citationHtml": "Rennes M&eacute;tropole, 2021, \"JDD avec GeoJson 2021-
↪07-13T10:23:46.409Z\", <a href=\"https://doi.org/10.5072/FK2/GIWCKB\" target=\"_blank\
↪">https://doi.org/10.5072/FK2/GIWCKB</a>, Root, DRAFT VERSION",
                "identifier_of_dataverse": "Sample_data",
                "name_of_dataverse": "Sample Data",
                "citation": "Métropole, 2021, \"JDD avec GeoJson 2021-07-13T10:23:46.
↪409Z\", https://doi.org/10.5072/FK2/GIWCKB, Root, DRAFT VERSION",
                "storageIdentifier": "file://10.5072/FK2/GIWCKB",
                "subjects": [
                    "Other"
                ],
                "fileCount": 0,
                "versionId": 9976,
                "versionState": "DRAFT",
                "createdAt": "2021-07-13T10:28:45Z",
                "updatedAt": "2021-07-13T10:28:45Z",
                "contacts": [
                    {
                        "name": "string",
                        "affiliation": "string"
                    }
                ],
                "metadataBlocks": {
                    "citation": {
                        "displayName": "Citation Metadata",
                        "fields": [
```


(continues on next page)

(continued from previous page)

```
                            {
                                "typeName": "dsDescription",
                                "multiple": true,
                                "typeClass": "compound",
                                "value": [
                                    {
                                        "dsDescriptionValue": {
                                            "typeName": "dsDescriptionValue",
                                            "multiple": false,
                                            "typeClass": "primitive",
                                            "value": "Démo sprint 5. Cette couche␣
↪représente l'emprise des cimetières sur le territoire des Métropole. Ces périmètres d
↪'emprise des cimetières sont issus du recensement des informations des PLU/POS de␣
↪chaque commune de la métropole, des données du cadastre DGFiP et d'un inventaire␣
↪terrain du Service Planification et Études Urbaines de Métropole"
                                        },
                                        "dsDescriptionDate": {
                                            "typeName": "dsDescriptionDate",
                                            "multiple": false,
                                            "typeClass": "primitive",
                                            "value": "2021-07-13"
                                        }
                                    }
                                ]
                            },
                            {
                                "typeName": "author",
                                "multiple": true,
                                "typeClass": "compound",
                                "value": [
                                    {
                                        "authorName": {
                                            "typeName": "authorName",
                                            "multiple": false,
                                            "typeClass": "primitive",
                                            "value": "Métropole"
                                        },
                                        "authorAffiliation": {
                                            "typeName": "authorAffiliation",
                                            "multiple": false,
                                            "typeClass": "primitive",
                                            "value": "string"
                                        }
                                    }
                                ]
                            }
                        ]
                    }
                },
                "authors": [
                    "Métropole"
                ]
```

(continues on next page)

(continued from previous page)


```
            },
            {
                "name": "Raja Ampat Islands",
                "type": "dataset",
                "url": "https://doi.org/10.5072/FK2/ITNXGR",
                "global_id": "doi:10.5072/FK2/ITNXGR",
                "description": "Raja Ampat is located off the northwest tip of Bird's␣
↪Head Peninsula on the island of New Guinea, in Indonesia's West Papua province, Raja␣
↪Ampat, or the Four Kings, is an archipelago comprising over 1,500 small islands, cays,␣
↪and shoals surrounding the four main islands of Misool, Salawati, Batanta, and Waigeo,␣
↪and the smaller island of Kofiau. The Raja Ampat archipelago straddles the Equator and␣
↪forms part of Coral Triangle which contains the richest marine biodiversity on earth.␣
↪Administratively, the archipelago is part of the province of West Papua (formerly␣
↪known as Irian Jaya). Most of the islands constitute the Raja Ampat Regency, which was␣
↪separated out from Sorong Regency in 2004. The regency encompasses around 70,000␣
↪square kilometres (27,000 sq mi) of land and sea, and has a population of about 50,000␣
↪(as of 2017). (Wikipedia: https://en.wikipedia.org/wiki/Raja_Ampat_Islands)",
                "published_at": "2020-07-30T09:23:34Z",
                "publisher": "Root",
                "citationHtml": "Admin, Dataverse, 2020, \"Raja Ampat Islands\", <a␣
↪href=\"https://doi.org/10.5072/FK2/ITNXGR\" target=\"_blank\">https://doi.org/10.5072/
↪FK2/ITNXGR</a>, Root, V1",
                "identifier_of_dataverse": "root",
                "name_of_dataverse": "Root",
                "citation": "Admin, Dataverse, 2020, \"Raja Ampat Islands\", https://doi.
↪org/10.5072/FK2/ITNXGR, Root, V1",
                "authors": [
                    "Admin, Dataverse"
                ]
            },
            {
                "name": "Sample Media Test",
                "type": "dataverse",
                "url": "https://68b2d8bb37c6/dataverse/Sample_media_test",
                "identifier": "Sample_media_test",
                "description": "Dataverse de test contenant les médias de Sample, comme␣
↪les images des fournisseurs et des producteurs",
                "published_at": "2021-04-08T15:04:14Z"
            },
            {
                "name": "Sample Test",
                "type": "dataverse",
                "url": "https://68b2d8bb37c6/dataverse/Sample_test",
                "identifier": "Sample_test",
                "description": "Dataverse utilisé pour les tests unitaires de Sample",
                "published_at": "2021-03-16T08:11:54Z"
            }
        ],
        "count_in_response": 4,
        "total_count_per_object_type": {
            "Datasets": 2,
            "Dataverses": 2
```


(continues on next page)

(continued from previous page)

```
        }
    }
}
```

5.4.4 Date Range Search Example

Below is an example of searching across a date range of Dataverse collections, datasets, and files that were published in 2018.

https://demo.dataverse.org/api/search?q=*&per_page=1000&sort=date&order=asc&q=*&fq=dateSort:[2018-01-01T00:00:00Z+TO+2019-01-01T00:00:00Z]

5.4.5 Iteration

Be default, up to 10 results are returned with every request (though this can be increased with the `per_page` parameter). To iterate through many results, increase the `start` parameter on each iteration until you reach the `total_count` in the response. An example in Python is below.

```
#!/usr/bin/env python
import urllib2
import json
base = 'https://demo.dataverse.org'
rows = 10
start = 0
page = 1
condition = True # emulate do-while
while (condition):
    url = base + '/api/search?q=*' + "&start=" + str(start)
    data = json.load(urllib2.urlopen(url))
    total = data['data']['total_count']
    print "=== Page", page, "==="
    print "start:", start, " total:", total
    for i in data['data']['items']:
        print "- ", i['name'], "(" + i['type'] + ")"
    start = start + rows
    page += 1
    condition = start < total
```

Output from iteration example

```
=== Page 1 ===
start: 0  total: 12
-  Spruce Goose (dataset)
-  trees.png (file)
-  Spruce (dataverse)
-  Trees (dataverse)
-  Darwin's Finches (dataset)
-  Finches (dataverse)
-  Birds (dataverse)
-  Rings of Conifers (dataset)
-  Chestnut Trees (dataverse)
```

(continues on next page)

(continued from previous page)

```
-  Sparrows (dataverse)
=== Page 2 ===
start: 10  total: 12
-  Chestnut Sparrows (dataverse)
-  Wrens (dataverse)
```

5.4.6 Search Services

Discovering Available Search Services

To discover available search services and their capabilities, you can use the Search Services API endpoint. Note: Configurable Search Services are an optional, experimental feature than may evolve faster than other parts of Dataverse.

Example API endpoint: https://demo.dataverse.org/api/search/services

This endpoint returns a list of available search services, including their names, and display names. It also indicates the default search service.

Example response:

```
{
    "status": "OK",
    "data": {
     "services": [
        {
            "name": "solr",
            "displayName": "Solr Search",
        },
        {
            "name": "externalSearch",
            "displayName": "External Search for Datasets",
        }
    ],
    "defaultService": "solr"
}
```

You can use the `name` values returned by this endpoint in the `search_service` parameter of the main search API to specify which search service to use for a particular query.

Using Different Search Services

To use a specific search service, include the `search_service` parameter in your search query and pass the `name`. For example:

https://demo.dataverse.org/api/search?q=trees&search_service=externalSearch

This query will use the `externalSearch` service (assuming it exists) instead of the default search service (`solr`).

> **Note**
>
> Other search services may not be complete replacements for the included `solr` service. For example, they may not support searching for collections or files (just datasets).

Developing Search Services

See *Search Services* in the Developer Guide.

5.5 Data Access API

The Data Access API provides programmatic download access to the files stored in a Dataverse installation. More advanced features of the Access API include format-specific transformations (thumbnail generation/resizing for images; converting tabular data into alternative file formats) and access to the data-level metadata that describes the contents of the tabular files.

Contents:

- *Downloading All Files in a Dataset*
  - *Basic Download By Dataset*
  - *Download By Dataset By Version*
- *Basic File Access*
  - *Parameters:*
  - *Headers:*
  - *Examples*
  - *Using a Preview URL Token*
- *Multiple File ("bundle") download*
  - *Parameters:*
- *"All Formats" bundled download for Tabular Files.*
  - *Parameters:*
- *Citation - Get Citation In Other Formats*
- *Data Variable Metadata Access*
  - *Parameters:*
- *Preprocessed Data*
- *Authentication and Authorization*
- *Access Requests and Processing*
  - *Allow Access Requests:*
  - *Request Access:*
  - *Grant File Access:*
  - *Reject File Access:*
  - *Revoke File Access:*
  - *List File Access Requests:*
  - *User Has Requested Access to a File:*

- *Get User Permissions on a File:*

5.5.1 Downloading All Files in a Dataset

The "download by dataset" API downloads as many files as possible from a dataset as a zipped bundle.

By default, tabular files are downloaded in their "archival" form (tab-separated values). To download the original files (Stata, for example), add `format=original` as a query parameter.

There are a number of reasons why not all of the files can be downloaded:

- Some of the files are restricted and your API token doesn't have access (you will still get the unrestricted files).
- The Dataverse installation has limited how large the zip bundle can be.

In the curl example below, the flags `-O` and `-J` are used. When there are no errors, this has the effect of saving the file under the name suggested by Dataverse (which as of v6.7 will be based on the persistent identifier of the dataset and the latest version number, for example `doi-10.70122-fk2-n2xgbj_1.1.zip`; in prior versions the file name was `dataverse_files.zip` in all cases). This mirrors the way the files are saved when downloaded in a browser. The flags also force error messages to be downloaded as a file.

Please note that in addition to the files from dataset, an additional file call "MANIFEST.TXT" will be included in the zipped bundle. It has additional information about the files.

There are two forms of the "download by dataset" API, a basic form and one that supports dataset versions.

Basic Download By Dataset

The basic form downloads files from the latest accessible version of the dataset. If you are not using an API token, this means the most recently published version. If you are using an API token with full access to the dataset, this means the draft version or the most recently published version if no draft exists.

Note

Files that require a Guestbook Response will require an additional step to supply the Guestbook Response. A POST to the same endpoint with the Guestbook Response in the body can return a signed url that can be used to download the file(s) via a browser or download manager. To determine what information is required in the response you must first get the Guestbook for the Dataset whose file(s) you are trying to access along with any Custom Questions. Call `GET http://$SERVER/api/guestbooks/{id}` with the guestbookId from the Dataset (See *Get JSON Representation of a Dataset* to get the Dataset with guestbookId) to retrieve the Guestbook and Custom Questions. Build the JSON response with the information requested and add it to the body of the POST call within "guestbookResponse":{}. For more about guestbooks, see *Dataset Guestbooks* in the User Guide.

A curl example using a DOI (no version):

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.70122/FK2/N2XGBJ

curl -L -O -J -H "X-Dataverse-key:$API_TOKEN" $SERVER_URL/api/access/dataset/
↪:persistentId/?persistentId=$PERSISTENT_ID
```

The fully expanded example above (without environment variables) looks like this:

```
curl -L -O -J -H X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx https://demo.
↪dataverse.org/api/access/dataset/:persistentId/?persistentId=doi:10.70122/FK2/N2XGBJ
```

Download By Dataset By Version

The second form of the "download by dataset" API allows you to specify which version you'd like to download files from. As with the `datasets` API endpoints described in the *Native API* section, the following identifiers can be used.

- `:draft` the draft version, if any
- `:latest` either a draft (if exists) or the latest published version.
- `:latest-published` the latest published version
- `x.y` a specific version, where `x` is the major version number and `y` is the minor version number.
- `x` same as `x.0`

Note

Files that require a Guestbook Response will require an additional step to supply the Guestbook Response. A POST to the same endpoint with the Guestbook Response in the body can return a signed url that can be used to download the file(s) via a browser or download manager. To determine what information is required in the response you must first get the Guestbook for the Dataset whose file(s) you are trying to access along with any Custom Questions. Call `GET http://$SERVER/api/guestbooks/{id}` with the guestbookId from the Dataset (See *Get JSON Representation of a Dataset* to get the Dataset with guestbookId) to retrieve the Guestbook and Custom Questions. Build the JSON response with the information requested and add it to the body of the POST call within "guestbookResponse":{}. For more about guestbooks, see *Dataset Guestbooks* in the User Guide.

A curl example using a DOI (with version):

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.70122/FK2/N2XGBJ
export VERSION=2.0

curl -O -J -H "X-Dataverse-key:$API_TOKEN" $SERVER_URL/api/access/dataset/:persistentId/
↪versions/$VERSION?persistentId=$PERSISTENT_ID
```

Similarly to the API above, this will save the downloaded bundle under the name based on the persistent identifier and the version number, for example, `doi-10.70122-fk2-n2xgbj_1.1.zip` or `doi-10.70122-fk2-n2xgbj_draft.zip`.

The fully expanded example above (without environment variables) looks like this:

```
curl -O -J -H X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx https://demo.
↪dataverse.org/api/access/dataset/:persistentId/versions/2.0?persistentId=doi:10.70122/
↪FK2/N2XGBJ
```

5.5.2 Basic File Access

Basic access URI:

`/api/access/datafile/$id`

Note

Files can be accessed using persistent identifiers. This is done by passing the constant `:persistentId` where the numeric id of the file is expected, and then passing the actual persistent id as a query parameter with the name `persistentId`. However, this file access method is only effective when the FilePIDsEnabled option is enabled, which can be authorized by the admin. For further information, refer to *:FilePIDsEnabled*.

Example: Getting the file whose DOI is *10.5072/FK2/J8SJZB*

```
GET http://$SERVER/api/access/datafile/:persistentId?persistentId=doi:10.5072/FK2/
↪J8SJZB
```

Note

Files that require a Guestbook Response will require an additional step to supply the Guestbook Response. A POST to the same endpoint with the Guestbook Response in the body can return a signed url that can be used to download the file(s) via a browser or download manager. For more about guestbooks, see *Dataset Guestbooks* in the User Guide. In the following JSON example please note that the *name*, *email*, *institution*, and *position* fields will default to the User's account information if not included in the response. Call `GET http://$SERVER/api/guestbooks/{id}` with the guestbookId from the Dataset (See *Get JSON Representation of a Dataset* to get the Dataset with guestbookId) to retrieve the Guestbook and Custom Questions. Build the JSON response with the information requested and add it to the body of the POST call within "guestbookResponse":{}.

Example

```
curl -X POST -H 'Content-Type:application/json' -H "X-Dataverse-key:$API_TOKEN" "
↪$SERVER_URL/api/access/datafile/:persistentId?persistentId=doi:10.5072/FK2/J8SJZB" -
↪d '{"guestbookResponse": {"name": "My Name", "email": "myemail@example.com",
↪"institution": "Harvard","position": "Staff", "answers": [{"id": 123,"value": "Good
↪"},{"id": 124,"value": ["Multi","Line"]},{"id": 125,"value": "Yellow"}]}}'
```

Parameters:

`format`

the following parameter values are supported (for tabular data files only):

Valu	Description
orig-i-nal	"Saved Original", the proprietary (SPSS, Stata, R, etc.) file from which the tabular data was ingested;
RDat	Tabular data as an R Data frame (generated; unless the "original" file was in R);
prep	"Pre-processed data", in JSON.
sub-set	Column-wise subsetting. You must also supply a comma separated list of variables in the "variables" query parameter. In this example, 123 and 127 are the database ids of data variables that belong to the data file with the id 6: `curl 'http://localhost:8080/api/access/datafile/6?format=subset&variables=123,127'`.

`noVarHeader`

(supported for tabular data files only; ignored for all other file types)

Value	Description
true|1	Tab-delimited data file, without the variable name header (added to tab. files by default)

`imageThumb`

the following parameter values are supported (for image and pdf files only):

Value	Description
true	Generates a thumbnail image by rescaling to the default thumbnail size (64 pixels wide).
`N`	Rescales the image to `N` pixels wide. `imageThumb=true` and `imageThumb=64` are equivalent.

Headers:

Header	Description
Range	Download a specified byte range. Examples: • `bytes=0-9` gets the first 10 bytes. • `bytes=10-19` gets 10 bytes from the middle. • `bytes=-10` gets the last 10 bytes. • `bytes=9-` gets all bytes except the first 10. Only a single range is supported. The "If-Range" header is not supported. For more on the "Range" header, see https://developer.mozilla.org/en-US/docs/Web/HTTP/Range_requests

Examples

A curl example of using the `Range` header to download the first 10 bytes of a file using its file id (database id):

```
export SERVER_URL=https://demo.dataverse.org
export FILE_ID=42
export RANGE=0-9

curl -H "Range:bytes=$RANGE" $SERVER_URL/api/access/datafile/$FILE_ID
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "Range:bytes=0-9" https://demo.dataverse.org/api/access/datafile/42
```

Using a Preview URL Token

If you want to access a file in a dataset shared by a preview URL (see *Preview URL to Review Unpublished Dataset*, *Get Dataset By Preview URL Token* and *Create a Preview URL for a Dataset*) using the API, you can use the token in the preview URL as API_TOKEN.

```
SERVER_URL=https://demo.dataverse.org
PREVIEW_URL_TOKEN=a56444bc-7697-4711-8964-e0577f055fd2
FILE_ID=1111111
FILENAME=example.txt
curl -H "X-Dataverse-key:$PREVIEW_URL_TOKEN" -o "$FILENAME" "$SERVER_URL/api/access/
↪datafile/$FILE_ID"
```

5.5.3 Multiple File ("bundle") download

`/api/access/datafiles/$id1,$id2,...$idN`

Alternate Form: POST to `/api/access/datafiles` with a `fileIds` input field containing the same comma separated list of file ids. This is most useful when your list of files surpasses the allowed URL length (varies but can be ~2000 characters).

Returns the files listed, zipped. As of v6.7 the name of the zipped bundle will be based on the persistent identifier of the parent dataset, for example, `doi-10.70122-fk2-xxyyzz.zip`; in prior versions the file name was `dataverse_files.zip` in all cases).

Note

If the request can only be completed partially - if only *some* of the requested files can be served (because of the permissions and/or size restrictions), the file MANIFEST.TXT included in the zipped bundle will have entries specifying the reasons the missing files could not be downloaded. IN THE FUTURE the API will return a 207 status code to indicate that the result was a partial success. (As of writing this - v.4.11 - this hasn't been implemented yet)

Note

If any of the datafiles have the `DirectoryLabel` attributes in the corresponding `FileMetadata` entries, these will be added as folders to the Zip archive, and the files will be placed in them accordingly.

Note

If Guestbook Responses are required they can be included in the body along with the file ids as JSON: `{"fileIds":[1,2,3], {"guestbookResponse": {"answers": []}}}`. For more about guestbooks, see *Dataset Guestbooks* in the User Guide.

Parameters:

`format` the following parameter values are supported (for tabular data files only):

Value	Description
original	"Saved Original", the proprietary (SPSS, Stata, R, etc.) file from which the tabular data was ingested;

5.5.4 "All Formats" bundled download for Tabular Files.

`/api/access/datafile/bundle/$id`

This is a convenience packaging method available for tabular data files. It returns a zipped bundle that contains the data in the following formats:

- Tab-delimited;
- "Saved Original", the proprietary (SPSS, Stata, R, etc.) file from which the tabular data was ingested;
- Generated R Data frame (unless the "original" above was in R);
- Data (Variable) metadata record, in DDI XML;
- File citation, in Endnote and RIS formats.

Parameters:

`fileMetadataId`

Value	Description
ID	Exports file with specific file metadata `ID`.

5.5.5 Citation - Get Citation In Other Formats

Dataverse can generate datafile citations in "EndNote", "RIS", "BibTeX", and "CSL" formats. This API call sends the raw format with the appropriate content-type (EndNote is XML, RIS and BibTeX are plain text, and CSL is JSON). ("Internal" is also a valid value, returning the content as HTML). This API call requires a format in the API call which can be any of the values listed above.

Usage example:

```
export SERVER_URL=https://demo.dataverse.org
export DATAFILE_ID=99
export FORMAT=EndNote
export VERSION=1.0
```

(continues on next page)

(continued from previous page)

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/access/datafile/$DATAFILE_ID/
↪citation/$FORMAT?version=$VERSION"
```

The `version` query parameter is optional. It will default to `:latest`. See *Dataset Version Specifiers* for the list of possible values. If you request a draft, an API token with access must be provided.

5.5.6 Data Variable Metadata Access

These methods are only available for tabular data files. (i.e., data files with associated data table and variable objects).

`/api/access/datafile/$id/metadata/ddi`

In its basic form the verb above returns a DDI fragment that describes the file and the data variables in it.

The DDI returned will only have two top-level sections:

- a single `fileDscr`, with the basic file information plus the numbers of variables and observations and the UNF of the file.
- a single `dataDscr` section, with one `var` section for each variable.

Example:

`http://localhost:8080/api/access/datafile/6/metadata/ddi`

```
<codeBook version="2.0">
   <fileDscr ID="f6">
      <fileTxt>
         <fileName>_73084.tab</fileName>
         <dimensns>
            <caseQnty>3</caseQnty>
            <varQnty>2</varQnty>
         </dimensns>
         <fileType>text/tab-separated-values</fileType>
      </fileTxt>
      <notes level="file" type="VDC:UNF" subject="Universal Numeric Fingerprint">
↪UNF:6:zChnyI3fjwNP+6qW0VryVQ==</notes>
   </fileDscr>
   <dataDscr>
      <var ID="v1" name="id" intrvl="discrete">
         <location fileid="f6"/>
         <labl level="variable">Personen-ID</labl>
         <sumStat type="mean">2.0</sumStat>
         <sumStat type="mode">.</sumStat>
         <sumStat type="medn">2.0</sumStat>
         <sumStat type="stdev">1.0</sumStat>
         <sumStat type="min">1.0</sumStat>
         <sumStat type="vald">3.0</sumStat>
         <sumStat type="invd">0.0</sumStat>
         <sumStat type="max">3.0</sumStat>
         <varFormat type="numeric"/>
         <notes subject="Universal Numeric Fingerprint" level="variable" type="VDC:UNF">
```

(continues on next page)

(continued from previous page)

```
↪UNF:6:AvELPR5QTaBbnq6S22Msow==</notes>
      </var>
      <var ID="v3" name="sex" intrvl="discrete">
         <location fileid="f6"/>
         <labl level="variable">Geschlecht</labl>
         <sumStat type="mean">1.3333333333333333</sumStat>
         <sumStat type="max">2.0</sumStat>
         <sumStat type="vald">3.0</sumStat>
         <sumStat type="mode">.</sumStat>
         <sumStat type="stdev">0.5773502691896257</sumStat>
         <sumStat type="invd">0.0</sumStat>
         <sumStat type="medn">1.0</sumStat>
         <sumStat type="min">1.0</sumStat>
         <catgry>
            <catValu>1</catValu>
            <labl level="category">Mann</labl>
         </catgry>
         <catgry>
            <catValu>2</catValu>
            <labl level="category">Frau</labl>
         </catgry>
         <varFormat type="numeric"/>
         <notes subject="Universal Numeric Fingerprint" level="variable" type="VDC:UNF">
↪UNF:6:XqQaMwOA63taX1YyBzTZYQ==</notes>
      </var>
   </dataDscr>
</codeBook>
```

Parameters:

`fileMetadataId`

Valu	Description
ID	Exports file with specific file metadata `ID`. For example for data file with id 6 and file metadata id 2: `curl 'http://localhost:8080/api/access/datafile/6/metadata/ddi?fileMetadataId=2'`

More information on DDI is available in the *Tabular Data, Representation, Storage and Ingest* section of the User Guide.

Advanced options/Parameters:

It is possible to request only specific subsets of, rather than the full file-level DDI record. This can be a useful optimization, in cases such as when an application needs to look up a single variable; especially with data files with large numbers of variables. See `variables=123,127` in the example above.

5.5.7 Preprocessed Data

`/api/access/datafile/$id?format=prep`

This method provides the "preprocessed data" - a summary record that describes the values of the data vectors in the tabular file, in JSON. These metadata values are used by earlier versions of Data Explorer, an external tool that integrates with a Dataverse installation (see *External Tools*). Please note that this format might change in the future.

5.5.8 Authentication and Authorization

Data Access API supports both session- and API key-based authentication.

If a session is available, and it is already associated with an authenticated user, it will be used for access authorization. If not, or if the user in question is not authorized to access the requested object, an attempt will be made to authorize based on an API key, if supplied. All of the API verbs above support the key parameter `key=...` as well as the newer `X-Dataverse-key` header. For more details, see "Authentication" in the *Introduction* section.

5.5.9 Access Requests and Processing

All of the following endpoints take the persistent identifier as a parameter in place of 'id'.

Allow Access Requests:

Allow or disallow users from requesting access to restricted files in a dataset where id is the database id of the dataset or pid is the persistent id (DOI or Handle) of the dataset to update.

A curl example using an `id`:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X PUT -d true http://$SERVER/api/access/{id}/
↪allowAccessRequest
```

A curl example using a `pid`:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X PUT -d true http://$SERVER/api/access/
↪:persistentId/allowAccessRequest?persistentId={pid}
```

Request Access:

`/api/access/datafile/$id/requestAccess`

This method requests access to the datafile whose id is passed on the behalf of an authenticated user whose key is passed. Note that not all datasets allow access requests to restricted files.

A curl example using an `id`:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X PUT http://$SERVER/api/access/datafile/{id}/
↪requestAccess
```

Note

Some installations of Dataverse may require you to provide a Guestbook Response when requesting access to certain restricted files. The response can be passed in the body of this call. See "Get a Guestbook for a Dataverse Collection" in the *Native API*.

Grant File Access:

`/api/access/datafile/{id}/grantAccess/{identifier}`

This method grants access to the datafile whose id is passed on the behalf of an authenticated user whose user identifier is passed with an @ prefix. The key of a user who can manage permissions of the datafile is required to use this method.

A curl example using an `id`:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X PUT http://$SERVER/api/access/datafile/{id}/
↪grantAccess/{@userIdentifier}
```

Reject File Access:

`/api/access/datafile/{id}/rejectAccess/{identifier}`

This method rejects the access request to the datafile whose id is passed on the behalf of an authenticated user whose user identifier is passed with an @ prefix. The key of a user who can manage permissions of the datafile is required to use this method.

A curl example using an `id`:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X PUT http://$SERVER/api/access/datafile/{id}/
↪rejectAccess/{@userIdentifier}
```

Revoke File Access:

`/api/access/datafile/{id}/revokeAccess/{identifier}`

This method revokes previously granted access to the datafile whose id is passed on the behalf of an authenticated user whose user identifier is passed with an @ prefix. The key of a user who can manage permissions of the datafile is required to use this method.

A curl example using an `id`:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE http://$SERVER/api/access/datafile/{id}/
↪revokeAccess/{@userIdentifier}
```

List File Access Requests:

`/api/access/datafile/{id}/listRequests`

This method returns a list of Authenticated Users who have requested access to the datafile whose id is passed. The key of a user who can manage permissions of the datafile is required to use this method.

A curl example using an `id`:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X GET $SERVER/api/access/datafile/{id}/listRequests
```

Query parameters have been added to retrieve the historical list of "created", "granted", and "rejected" requests:

- *includeHistory* When *true* this will force the return of all requests and not just the "created" ones.
- *start* For pagination, use this to request a specific page.
- *per_page* For pagination, use this to limit the number of items in each paged list.

> **Note**
>
> Pagination is only available when *includeHistory* is *true*

If requesting a page beyond the last page this API will return a 404 "There are no access requests for this file:..." If requesting a page before page 1 or requesting the number of items to be 0 or less this API will ignore these parameters and return the entire list.

A curl example using an `id`:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER/api/access/datafile/{id}/
↪listRequests?includeHistory=true&start=1&per_page=20"
```

User Has Requested Access to a File:

`/api/access/datafile/{id}/userFileAccessRequested`

This method returns true or false depending on whether or not the calling user has requested access to a particular file.

A curl example using an `id`:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER/api/access/datafile/{id}/
↪userFileAccessRequested"
```

Get User Permissions on a File:

`/api/access/datafile/{id}/userPermissions`

This method returns the permissions that the calling user has on a particular file.

In particular, the user permissions that this method checks, returned as booleans, are the following:

- Can download the file
- Can manage the file permissions
- Can edit the file owner dataset

A curl example using an `id`:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X GET "http://$SERVER/api/access/datafile/{id}/
↪userPermissions"
```

Authentication is required for this API.

5.6 Native API

The Dataverse Software exposes most of its GUI functionality via a REST-based API. This section describes that functionality. Most API endpoints require an API token that can be passed as the `X-Dataverse-key` HTTP header or in the URL as the `key` query parameter.

Note

Some API endpoint allow CORS (cross-origin resource sharing), which makes them usable from scripts running in web browsers. These endpoints are marked with a *CORS* badge.

Note

Bash environment variables shown below. The idea is that you can "export" these environment variables before copying and pasting the commands that use them. For example, you can set `$SERVER_URL` by running `export SERVER_URL="https://demo.dataverse.org"` in your Bash shell. To check if the environment variable was set properly, you can "echo" it (e.g. `echo $SERVER_URL`). See also *curl Examples and Environment Variables*.

Warning

The Dataverse Software's API is versioned at the URI - all API calls may include the version number like so: `https://server-address/api/v1/...`. Omitting the `v1` part would default to the latest API version (currently 1). When writing scripts/applications that will be used for a long time, make sure to specify the API version, so they don't break when the API is upgraded.

Contents:

- *Dataverse Collections*
  - *Create a Dataverse Collection*
  - *Update a Dataverse Collection*
  - *View a Dataverse Collection*
  - *Delete a Dataverse Collection*
  - *Show Contents of a Dataverse Collection*
  - *Report the data (file) size of a Dataverse Collection*
  - *List Roles Defined in a Dataverse Collection*
  - *List the Allowed Metadata Languages of a Dataverse Collection*

- *Set the Allowed Metadata Language of a Dataverse Collection*
- *List Facets Configured for a Dataverse Collection*
- *Set Facets for a Dataverse Collection*
- *List Metadata Block Facets Configured for a Dataverse Collection*
- *List Field Type Input Levels Configured for a Dataverse Collection*
- *Set Metadata Block Facets for a Dataverse Collection*
- *Configure a Dataverse Collection to Inherit Its Metadata Block Facets from Its Parent*
- *Create a New Role in a Dataverse Collection*
- *List Role Assignments in a Dataverse Collection*
- *Assign Default Role to User Creating a Dataset in a Dataverse Collection*
- *Get Default Role Assigned to User Creating a Dataset in a Dataverse Collection*
- *Assign a New Role on a Dataverse Collection*
- *Delete Role Assignment from a Dataverse Collection*
- *List Metadata Blocks Defined on a Dataverse Collection*
- *Define Metadata Blocks for a Dataverse Collection*
- *Determine if a Dataverse Collection Inherits Its Metadata Blocks from Its Parent*
- *Configure a Dataverse Collection to Inherit Its Metadata Blocks from Its Parent*
- *Retrieve a Dataset JSON Schema for a Collection*
- *Validate Dataset JSON File for a Collection*
- *List Dataverse Collections to Which a Given Dataset or Dataverse Collection May Be Linked*
- *List Featured Collections for a Dataverse Collection*
- *Set Featured Collections for a Dataverse Collection*
- *Remove Featured Collections from a Dataverse Collection*
- *Get User Permissions on a Dataverse*
- *List Locally FAIR Role Assignees for a Dataverse Collection*
- *Set Locally FAIR Role Assignees for a Dataverse Collection*
- *Add a Locally FAIR Role Assignee to a Dataverse Collection*
- *Delete a Locally FAIR Role Assignee from a Dataverse Collection*
- *Create a Dataset in a Dataverse Collection*
  - *Submit Incomplete Dataset*
  - *Submit Dataset*
  - *Create a Dataset with a Dataset Type (Software, Workflow, Review, etc.)*
- *Import a Dataset into a Dataverse Collection*
  - *Import a Dataset with a Dataset Type (Software, Workflow, Review, etc.)*
- *Import a Dataset into a Dataverse Installation with a DDI file*

- *Publish a Dataverse Collection*
- *Retrieve Guestbook Responses for a Dataverse Collection*
- *Guestbooks*
  - *Create a Guestbook for a Dataverse Collection*
  - *Get a list of Guestbooks for a Dataverse Collection*
  - *Get a Guestbook for a Dataverse Collection*
  - *Update a Guestbook for a Dataverse Collection*
  - *Enable or Disable a Guestbook for a Dataverse Collection*
- *Retrieve Guestbook Responses for a Guestbook*
- *Change Collection Attributes*
- *Update Collection Input Levels*
- *Collection Storage Quotas*
- *Storage Quotas on Individual Datasets*
- *List All Collection Featured Items*
- *Update All Collection Featured Items*
- *Delete All Collection Featured Items*
- *Create a Collection Featured Item*
- *Update a Collection Featured Item*
- *Delete a Collection Featured Item*
- *Get a Collection Featured Item Image*
- *List Templates of a Collection*
- *List Single Template by its Identifier*
- *Delete a Template by its Identifier*
- *Create a Template for a Collection*
- *Update the Metadata and Instructions of a Template*
- *Update the License or Terms Of Use of a Template*
- *Update the Terms Of Access of a Template*
- *Set a Default Template for a Collection*
- *Remove a Default Template for a Collection*
- *Dataverse Role Assignment History*

- *Datasets*
  - *Dataset Version Specifiers*
  - *Get JSON Representation of a Dataset*
  - *List Versions of a Dataset*
  - *Get Version of a Dataset*

- *Export Metadata of a Dataset in Various Formats*
  - *Available Dataset Metadata Exporters*
  - *Schema.org JSON-LD*
- *List Files in a Dataset*
- *Get File Counts in a Dataset*
- *View Dataset Files and Folders as a Directory Index*
- *List All Metadata Blocks for a Dataset*
- *List Single Metadata Block for a Dataset*
- *Compare Versions of a Dataset*
- *Get Versions of a Dataset with Summary of Changes*
- *Update Metadata For a Dataset*
- *Edit Dataset Metadata*
- *Delete Dataset Metadata*
- *Update Dataset Terms of Access*
- *Publish a Dataset*
- *Delete Dataset Draft*
- *Deaccession Dataset*
- *Set Citation Date Field Type for a Dataset*
- *Revert Citation Date Field Type to Default for Dataset*
- *List Role Assignments in a Dataset*
- *Assign a New Role on a Dataset*
- *Delete Role Assignment from a Dataset*
- *Create a Preview URL for a Dataset*
- *Get the Preview URL for a Dataset*
- *Delete the Preview URL from a Dataset*
- *Add a File to a Dataset*
- *Add a Remote File to a Dataset*
- *Update File Metadata*
- *Cleanup Storage of a Dataset*
- *Adding Files To a Dataset via Other Tools*
- *Imposing a limit to the number of files allowed to be uploaded to a Dataset*
- *Setting the files count limit on a Dataset*
- *Report the data (file) size of a Dataset*
- *Get the size of Downloading all the files of a Dataset Version*
- *Get the Download count of a Dataset*

- *Submit a Dataset for Review*
- *Return a Dataset to Author*
- *Link a Dataset*
- *Dataset Locks*
  - *Manage Locks on a Specific Dataset*
  - *List Locks Across All Datasets*
- *Dataset Metrics*
  - *Retrieving Total Views for a Dataset*
  - *Retrieving Unique Views for a Dataset*
  - *Retrieving Total Downloads for a Dataset*
  - *Retrieving Unique Downloads for a Dataset*
  - *Retrieving Citations for a Dataset*
- *Delete Unpublished Dataset*
- *Delete Published Dataset*
- *Configure a Dataset to Use a Specific File Store*
- *View the Timestamps on a Dataset*
- *Set an Embargo on Files in a Dataset*
- *Remove an Embargo on Files in a Dataset*
- *Set a Retention Period on Files in a Dataset*
- *Remove a Retention Period on Files in a Dataset*
- *Get the Archival Status of a Dataset By Version*
- *Set the Archival Status of a Dataset By Version*
- *Delete the Archival Status of a Dataset By Version*
- *Get Dataset External Tool URL*
- *Get External Tool Parameters*
- *Retrieve Signposting Information*
  - *Signposting Link HTTP Header*
  - *Signposting Linkset API Endpoint*
- *Get Dataset By Preview URL Token*
- *Get Citation*
- *Get Citation In Other Formats*
- *Get Citation by Preview URL Token*
- *Get Summary Field Names*
- *Get Available Dataset File Categories*
- *Configure When a Dataset Guestbook Appears (If Enabled)*

  - *Get User Permissions on a Dataset*
  - *Know If a User Can Download at Least One File from a Dataset Version*
  - *Configure The PID Generator a Dataset Uses (If Enabled)*
  - *Reconcile the PID of a Dataset (If Multiple PID Providers Are Enabled)*
  - *Dataset Types*
    - *List Dataset Types*
    - *Get Dataset Type*
    - *Add Dataset Type*
    - *Delete Dataset Type*
    - *Link Dataset Type with Metadata Blocks*
    - *Set Available Licenses for a Dataset Type*
  - *Dataset Version Notes*
  - *Delete Files from a Dataset*
  - *Dataset Role Assignment History*
  - *Dataset Files Role Assignment History*
  - *Update Dataset License*
  - *List Reviews*
- *Files*
  - *Get JSON Representation of a File*
  - *Get JSON Representation of a File given a Dataset Version*
  - *Get JSON Representation of a file's versions*
  - *Adding Files*
  - *Accessing (downloading) files*
  - *Restrict Files*
  - *Uningest a File*
  - *Reingest a File*
  - *Redetect File Type*
  - *Extract NcML*
  - *Replacing Files*
  - *Deleting Files*
  - *Getting File Metadata*
  - *Getting File Data Tables*
  - *Getting File Download Count*
  - *File Has Been Deleted*
  - *Updating File Metadata*

    - *Updating File Metadata Categories*
    - *Updating File Tabular Tags*
    - *Editing Variable Level Metadata*
    - *Get File Citation as JSON*
    - *Provenance*
        * *Get Provenance JSON for an uploaded file*
        * *Get Provenance Description for an uploaded file*
        * *Create/Update Provenance JSON and provide related entity name for an uploaded file*
        * *Create/Update Provenance Description for an uploaded file*
        * *Delete Provenance JSON for an uploaded file*
    - *Datafile Integrity*
    - *Get File External Tool URL*
    - *Get File External Tool Parameters*
    - *Get External Tool Parameters*
    - *Get Fixity Algorithm*
- *Users Token Management*
    - *Find a Token's Expiration Date*
    - *Recreate a Token*
    - *Delete a Token*
- *Builtin Users*
    - *Create a Builtin User*
- *Roles*
    - *JSON Representation of a Role*
    - *Create Role*
    - *Show Role*
    - *Delete Role*
    - *Get User Selectable Roles*
- *Explicit Groups*
    - *Create New Explicit Group*
    - *List Explicit Groups in a Dataverse Collection*
    - *Show Single Group in a Dataverse Collection*
    - *Update Group in a Dataverse Collection*
    - *Delete Group from a Dataverse Collection*
    - *Add Multiple Role Assignees to an Explicit Group*
    - *Add a Role Assignee to an Explicit Group*

    - *Remove a Role Assignee from an Explicit Group*
- *Shibboleth Groups*
- *Info*
    - *Show Dataverse Software Version and Build Number*
    - *Show Dataverse Installation Server Name*
    - *Show Custom Popup Text for Publishing Datasets*
    - *Show Custom Popup Text for Submitting Datasets For Review*
    - *Show Disclaimer for Publishing Datasets*
    - *Show Disclaimer for Submitting For Review Datasets*
    - *Get Application Terms of Use (General Terms of Use)*
    - *Get API Terms of Use URL*
    - *Show Support Of Incomplete Metadata Deposition*
    - *Get Zip File Download Limit*
    - *Get Maximum Embargo Duration In Months*
    - *Get Dataset Metadata Export Formats*
    - *Get Customization File Contents*
- *Metadata Blocks*
    - *Show Info About All Metadata Blocks*
    - *Show Info About Single Metadata Block*
- *Dataset Fields*
    - *List All Facetable Dataset Fields*
    - *Set displayOnCreate for a Dataset Field*
- *Notifications*
    - *Get All Notifications by User*
    - *Get Unread Count*
    - *Mark Notification As Read*
    - *Delete Notification by User*
    - *Get All Muted In-app Notifications by User*
    - *Mute In-app Notification by User*
    - *Unmute In-app Notification by User*
    - *Get All Muted Email Notifications by User*
    - *Mute Email Notification by User*
    - *Unmute Email Notification by User*
- *User Information*
    - *Get User Information in JSON Format*

- *Managing Harvesting Server and Sets*
  - *List All Harvesting Sets*
  - *List A Specific Harvesting Set*
  - *Create a Harvesting Set*
  - *Modify an Existing Harvesting Set*
  - *Delete an Existing Harvesting Set*
- *Managing Harvesting Clients*
  - *List All Configured Harvesting Clients*
  - *Show a Specific Harvesting Client*
  - *Create a Harvesting Client*
  - *Modify a Harvesting Client*
  - *Delete a Harvesting Client*
  - *Harvesting from DataCite*
- *PIDs*
  - *Get Info on a PID*
  - *List Unreserved PIDs*
  - *Reserve a PID*
  - *Delete a PID*
  - *Get Information about Configured PID Providers*
  - *Get the id of the PID Provider Managing a Given PID*
- *Admin*
  - *Manage Database Settings*
    - *List All Database Settings*
    - *Get Single Database Setting*
    - *Get Single Database Setting With Language/Locale*
    - *Configure Single Database Setting*
    - *Configure Single Database Setting With Language/Locale*
    - *Configure All Database Settings*
    - *Delete Single Database Setting*
    - *Delete Single Database Setting With Language/Locale*
  - *List All Feature Flags*
  - *Show Feature Flag Status*
  - *Manage Banner Messages*
  - *List Authentication Provider Factories*
  - *List Authentication Providers*

- *Add Authentication Provider*
- *Show Authentication Provider*
- *Enable or Disable an Authentication Provider*
- *Check If an Authentication Provider is Enabled*
- *Delete an Authentication Provider*
- *List Global Roles*
- *Create Global Role*
- *Update Global Role*
- *Delete Global Role*
- *List Users*
- *List Single User*
- *Create an Authenticated User*
- *Merge User Accounts*
- *Change User Identifier*
- *Toggle Superuser Status*
- *Set Superuser Status*
- *Delete a User*
- *Deactivate a User*
- *Show User Traces*
- *Remove All Roles from a User*
- *List Role Assignments of a Role Assignee*
- *List Permissions a User Has on a Dataverse Collection or Dataset*
- *List Dataverse collections a user can act on based on their permissions*
- *Show Role Assignee*
- *Saved Search*
  - *List All Saved Searches*
  - *List a Saved Search by Database ID*
  - *Delete a Saved Search by Database ID*
  - *Execute a Saved Search and Make Links*
  - *Execute All Saved Searches and Make Links*
- *Dataset Integrity*
- *Datafile Integrity*
- *Physical Files Validation in a Dataset*
- *Update Checksums To Use New Algorithm*
- *Dataset Validation*

    - *Datafile Audit*
    - *Workflows*
    - *Metrics*
    - *Inherit Dataverse Collection Role Assignments*
    - *Manage Available Standard License Terms*
    - *List Dataset Templates*
    - *Delete Dataset Template*
    - *Request Signed URL*
    - *Send Feedback To Contact(s) Admin API*
    - *Send Feedback To Contact(s)*
    - *Reset Thumbnail Failure Flags*
    - *Download File from /tmp*
- *MyData*
    - *MyData Collection List*

5.6.1 Dataverse Collections

Create a Dataverse Collection

A Dataverse collection is a container for datasets and other Dataverse collections as explained in the *Dataverse Collection Management* section of the User Guide.

The steps for creating a Dataverse collection are:

- Prepare a JSON file containing the name, description, etc, of the Dataverse collection you'd like to create.
- Figure out the alias or database id of the "parent" Dataverse collection into which you will be creating your new Dataverse collection.
- Execute a curl command or equivalent.

Download `dataverse-complete.json` file and modify it to suit your needs. The fields `name`, `alias`, and `dataverseContacts` are required. The optional field `datasetFileCountLimit` can be used to set the maximum number of files a dataset can have. The optional field `guestbookRoot`, when set to true, will allow datasets in this collection to be assigned a Guestbook from a parent collection. The controlled vocabulary for `dataverseType` is the following:

- `DEPARTMENT`
- `JOURNALS`
- `LABORATORY`
- `ORGANIZATIONS_INSTITUTIONS`
- `RESEARCHERS`
- `RESEARCH_GROUP`
- `RESEARCH_PROJECTS`

- TEACHING_COURSES
- UNCATEGORIZED

```
{
  "name": "Scientific Research",
  "alias": "science",
  "dataverseContacts": [
    {
      "contactEmail": "pi@example.edu"
    },
    {
      "contactEmail": "student@example.edu"
    }
  ],
  "affiliation": "Scientific Research University",
  "description": "We do all the science.",
  "dataverseType": "LABORATORY",
  "datasetFileCountLimit": 10,
  "guestbookRoot": false
}
```

The curl command below assumes you have kept the name "dataverse-complete.json" and that this file is in your current working directory.

Next you need to figure out the alias or database id of the "parent" Dataverse collection into which you will be creating your new Dataverse collection. Out of the box the top level Dataverse collection has an alias of "root" and a database id of "1" but your installation may vary. The easiest way to determine the alias of your root Dataverse collection is to click "Advanced Search" and look at the URL. You may also choose a parent under the root.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PARENT=root

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/dataverses/$PARENT" --
↪upload-file dataverse-complete.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/dataverses/root" --upload-file dataverse-complete.json
```

You should expect an HTTP 200 response and JSON beginning with "status":"OK" followed by a representation of the newly-created Dataverse collection.

The request JSON supports an optional `metadataBlocks` object, with the following supported sub-objects:

- `metadataBlockNames`: The names of the metadata blocks you want to add to the Dataverse collection.
- `inputLevels`: The names of the fields in each metadata block for which you want to add a custom configuration regarding their inclusion or requirement when creating and editing datasets in the new Dataverse collection. Note

that if the corresponding metadata blocks names are not specified in the `metadataBlockNames` field, they will be added automatically to the Dataverse collection.

- `facetIds`: The names of the fields to use as facets for browsing datasets and collections in the new Dataverse collection. Note that the order of the facets is defined by their order in the provided JSON array.

To obtain an example of how these objects are included in the JSON file, download `dataverse-complete-optional-params.json` file and modify it to suit your needs.

Update a Dataverse Collection

Updates an existing Dataverse collection using a JSON file following the same structure as the one used in the API for the creation. (see *Create a Dataverse Collection*).

The steps for updating a Dataverse collection are:

- Prepare a JSON file containing the fields for the properties you want to update. You do not need to include all the properties, only the ones you want to update.
- Execute a curl command or equivalent.

As an example, you can download `dataverse-complete.json` file and modify it to suit your needs. The controlled vocabulary for `dataverseType` is the following:

- `DEPARTMENT`
- `JOURNALS`
- `LABORATORY`
- `ORGANIZATIONS_INSTITUTIONS`
- `RESEARCHERS`
- `RESEARCH_GROUP`
- `RESEARCH_PROJECTS`
- `TEACHING_COURSES`
- `UNCATEGORIZED`

The curl command below assumes you are using the name "dataverse-complete.json" and that this file is in your current working directory.

Next you need to figure out the alias or database id of the Dataverse collection you want to update.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export DV_ALIAS=dvAlias

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/dataverses/$DV_ALIAS" --
↪upload-file dataverse-complete.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/dataverses/dvAlias" --upload-file dataverse-complete.json
```

You should expect an HTTP 200 response and JSON beginning with "status":"OK" followed by a representation of the updated Dataverse collection.

Same as in *Create a Dataverse Collection*, the request JSON supports an optional `metadataBlocks` object, with the following supported sub-objects:

- `metadataBlockNames`: The names of the metadata blocks to be assigned to the Dataverse collection.
- `inputLevels`: The names of the fields in each metadata block for which you want to add a custom configuration regarding their inclusion or requirement when creating and editing datasets in the Dataverse collection. Note that if the corresponding metadata blocks names are not specified in the `metadataBlockNames` field, they will be added automatically to the Dataverse collection.
- `facetIds`: The names of the fields to use as facets for browsing datasets and collections in the Dataverse collection. Note that the order of the facets is defined by their order in the provided JSON array.

Note that setting any of these fields overwrites the previous configuration.

When it comes to omitting these fields in the JSON:

- Omitting `facetIds` or `metadataBlockNames` causes no change to the Dataverse collection. To delete the current configuration and inherit the corresponding configuration from its parent include the flag `inheritFacetsFromParent` and/or `inheritMetadataBlocksFromParent` respectively.
- Omitting `inputLevels` causes no change to the Dataverse collection. Including the flag `inheritMetadataBlocksFromParent` will cause the custom `inputLevels` to be deleted and inherited from the parent.
- Omitting the entire `metadataBlocks` object in the request JSON would cause no change to the `inputLevels`, `facetIds` or `metadataBlockNames` of the Dataverse collection.

To obtain an example of how these objects are included in the JSON file, download `dataverse-complete-optional-params.json` file and modify it to suit your needs.

To force the configurations to be deleted and inherited from the parent's configuration include the following `metadataBlocks` object in your JSON

```
"metadataBlocks": {
  "inheritMetadataBlocksFromParent": true,
  "inheritFacetsFromParent": true
}
```

Note

Including both the list `metadataBlockNames` and the flag `"inheritMetadataBlocksFromParent": true` will result in an error being returned {"status": "ERROR", "message": "Metadata block can not contain both metadataBlockNames and inheritMetadataBlocksFromParent: true"}. The same is true for `facetIds` and `inheritFacetsFromParent`.

See also *Change Collection Attributes*.

View a Dataverse Collection

View a JSON representation of the Dataverse collection identified by `$id`. `$id` can be the database ID of the Dataverse collection, its alias, or the special value `:root` for the root Dataverse collection.

To view a published Dataverse collection:

```
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl "$SERVER_URL/api/dataverses/$ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/dataverses/root"
```

If you want to include the Dataverse collections that this collection is part of, you must set `returnOwners` query parameter to `true`.

Usage example:

```
curl "https://demo.dataverse.org/api/dataverses/root?returnOwners=true"
```

If you want to include the child count of the Dataverse, which represents the number of dataverses, datasets, or files within the dataverse, you must set `returnChildCount` query parameter to `true`. Please note that this count is for direct children only. It doesn't count children of subdataverses.

Usage example:

```
curl "https://demo.dataverse.org/api/dataverses/root?returnChildCount=true"
```

If a user with EditDataverse permissions wants to ignore the setting `ExcludeEmailFromExport` in order to see the contact emails, they must include the `ignoreSettingExcludeEmailFromExport` query parameter.

Usage example:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root?ignoreSettingExcludeEmailFromExport=true"
```

To view an unpublished Dataverse collection:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root"
```

Delete a Dataverse Collection

Before you may delete a Dataverse collection you must first delete or move all of its contents elsewhere.

Deletes the Dataverse collection whose database ID or alias is given:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/dataverses/$ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/dataverses/root"
```

Show Contents of a Dataverse Collection

Lists all the Dataverse collections and datasets directly under a Dataverse collection (direct children only, not recursive) specified by database id or alias. If you pass your API token and have access, unpublished Dataverse collections and datasets will be included in the response. The list will be ordered by database id within type of object. That is, all Dataverse collections will be listed first and ordered by database id, then all datasets will be listed ordered by database id.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/contents"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/contents"
```

Report the data (file) size of a Dataverse Collection

Shows the combined size in bytes of all the files uploaded into the Dataverse collection `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/storagesize"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/storagesize"
```

The size of published and unpublished files will be summed both in the Dataverse collection specified and beneath all its sub-collections, recursively. By default, only the archival files are counted - i.e., the files uploaded by users (plus the tab-delimited versions generated for tabular data files on ingest). If the optional argument `includeCached=true` is specified, the API will also add the sizes of all the extra files generated and cached by the Dataverse installation - the resized thumbnail versions for image files, the metadata exports for published datasets, etc.

List Roles Defined in a Dataverse Collection

All the roles defined directly in the Dataverse collection identified by `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/roles"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/roles"
```

List the Allowed Metadata Languages of a Dataverse Collection

Shows the allowed metadata languages of the Dataverse collection `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/
↪allowedMetadataLanguages"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/allowedMetadataLanguages"
```

If there are no metadata languages configured on the server, this call returns an empty array. If the Dataverse collection has a mandatory metadata language, the return value is an array of that single language, otherwise it's an array of all available metadata languages on the server.

Set the Allowed Metadata Language of a Dataverse Collection

Sets the allowed metadata language of the Dataverse collection `id` to `langCode` if it's available on the server:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root
export LANGCODE=en

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/dataverses/$ID/
↪allowedMetadataLanguages/$LANGCODE"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/dataverses/root/allowedMetadataLanguages/en"
```

Returns an array of the set metadata language. If the metadata language is not available on the server, this call responds with a 400 BAD REQUEST.

List Facets Configured for a Dataverse Collection

List all the facets for a given Dataverse collection `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/facets"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/facets"
```

By default, this endpoint will return an array including the facet names. If more detailed information is needed, we can set the query parameter `returnDetails` to `true`, which will return the display name and id in addition to the name for each facet:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/facets?
↪returnDetails=true"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/facets?returnDetails=true"
```

Set Facets for a Dataverse Collection

Assign search facets for a given Dataverse collection identified by `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/dataverses/$ID/facets" --
↪upload-file dataverse-facets.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/dataverses/root/facets" --upload-file dataverse-facets.json
```

Where `dataverse-facets.json` contains a JSON encoded list of metadata keys (e.g. `["authorName", "authorAffiliation"]`).

List Metadata Block Facets Configured for a Dataverse Collection

List the metadata block facet configuration with all the metadata block configured for a given Dataverse collection `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/metadatablockfacets"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/metadatablockfacets"
```

List Field Type Input Levels Configured for a Dataverse Collection

List the dataverse field type input levels configured for a given Dataverse collection `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/inputLevels"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/inputLevels"
```

Set Metadata Block Facets for a Dataverse Collection

Sets the metadata blocks that will appear in the `Dataset Features` facet category for a given Dataverse collection identified by `id`.

In order to set or clear the metadata blocks for a collection, you must first *set the metadata block facet root to true*.

To clear the metadata blocks set by a parent collection, submit an empty array (e.g. `[]`):

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -X POST -H "Content-type:application/json" "$SERVER_
↪URL/api/dataverses/$ID/metadatablockfacets" --upload-file metadata-block-facets.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST -H "Content-
↪type:application/json" "https://demo.dataverse.org/api/dataverses/root/
↪metadatablockfacets" --upload-file metadata-block-facets.json
```

Where `metadata-block-facets.json` contains a JSON encoded list of metadata block names (e.g. `["socialscience","geospatial"]`). This endpoint supports an empty list (e.g. `[]`)

Configure a Dataverse Collection to Inherit Its Metadata Block Facets from Its Parent

Set whether the Dataverse collection is a metadata block facet root, or does it uses its parent metadata block facets. Possible values are `true` and `false` (both are valid JSON expressions).

When updating the root to false, it will clear any metadata block facets from the collection. When updating to true, it will copy the metadata block facets from the parent collection:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -X POST -H "Content-type:application/json" "$SERVER_
↪URL/api/dataverses/$ID/metadatablockfacets/isRoot" -d 'true'
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST -H "Content-
↪type:application/json" "https://demo.dataverse.org/api/dataverses/root/
↪metadatablockfacets/isRoot" -d 'true'
```

Create a New Role in a Dataverse Collection

Creates a new role under Dataverse collection `id`. Needs a json file with the role description:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -H "Content-type:application/json" -X POST "$SERVER_
↪URL/api/dataverses/$ID/roles" --upload-file roles.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Content-
↪type:application/json" -X POST "https://demo.dataverse.org/api/dataverses/root/roles" -
↪-upload-file roles.json
```

For `roles.json` see *JSON Representation of a Role*

> **Note**
>
> Only a Dataverse installation account with superuser permissions is allowed to create roles in a Dataverse Collection.

List Role Assignments in a Dataverse Collection

List all the role assignments at the given Dataverse collection:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/assignments"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/assignments"
```

Assign Default Role to User Creating a Dataset in a Dataverse Collection

Assign a default role to a user creating a dataset in a Dataverse collection `id` where `roleAlias` is the database alias of the role to be assigned (requires `ManageDataversePermissions`):

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root
export ROLE_ALIAS=curator

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/dataverses/$ID/
↪defaultContributorRole/$ROLE_ALIAS"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/dataverses/root/defaultContributorRole/curator"
```

Note: You may use "none" as the `ROLE_ALIAS`. This will prevent a user who creates a dataset from having any role on that dataset. It is not recommended for Dataverse collections with human contributors.

Get Default Role Assigned to User Creating a Dataset in a Dataverse Collection

Get the default role that is assigned to a user creating a dataset in a Dataverse collection `id` (requires `ManageDataversePermissions`):

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/
↪defaultContributorRole"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/defaultContributorRole"
```

Assign a New Role on a Dataverse Collection

Assigns a new role, based on the POSTed JSON:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -X POST -H "Content-Type: application/json" "
↪$SERVER_URL/api/dataverses/$ID/assignments" --upload-file role.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST -H "Content-Type:␣
↪application/json" "https://demo.dataverse.org/api/dataverses/root/assignments" --
↪upload-file role.json
```

POSTed JSON example (the content of `role.json` file):

```
{
  "assignee": "@uma",
  "role": "curator"
}
```

Delete Role Assignment from a Dataverse Collection

Delete the assignment whose id is `$id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root
export ASSIGNMENT_ID=6

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/dataverses/$ID/
↪assignments/$ASSIGNMENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/dataverses/root/assignments/6"
```

List Metadata Blocks Defined on a Dataverse Collection

Get the metadata blocks defined on a Dataverse collection which determine which field are available to authors when they create and edit datasets within that Dataverse collection. This feature is described in *General Information* section of Dataverse Collection Management of the User Guide.

Please note that an API token is only required if the Dataverse collection has not been published.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/metadatablocks"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/metadatablocks"
```

This endpoint supports the following optional query parameters:

- `returnDatasetFieldTypes`: Whether or not to return the dataset field types present in each metadata block. If not set, the default value is false.
- `onlyDisplayedOnCreate`: Whether or not to return only the metadata blocks that are displayed on dataset creation. If `returnDatasetFieldTypes` is true, only the dataset field types shown on dataset creation will be returned within each metadata block. If not set, the default value is false.
- `datasetType`: Optionally return additional fields from metadata blocks that are linked with a particular dataset type (see *Dataset Types* in the User Guide). Pass a single dataset type as a string. For a list of dataset types you can pass, see *List Dataset Types*.

An example using the optional query parameters is presented below:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root
export DATASET_TYPE=software

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/metadatablocks?
↪returnDatasetFieldTypes=true&onlyDisplayedOnCreate=true&datasetType=$DATASET_TYPE"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/metadatablocks?returnDatasetFieldTypes=true&
↪onlyDisplayedOnCreate=true&datasetType=software"
```

Define Metadata Blocks for a Dataverse Collection

You can define the metadata blocks available to authors within a Dataverse collection.

The metadata blocks that are available with a default Dataverse installation are in `define-metadatablocks.json` (also shown below) and you should download this file and edit it to meet your needs. Please note that the "citation" metadata block is required. You must have "EditDataverse" permission on the Dataverse collection.

```
[
  "citation",
  "geospatial",
  "socialscience",
  "astrophysics",
  "biomedical",
  "journal"
]
```

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/dataverses/$ID/
↪metadatablocks" -H \"Content-type:application/json\" --upload-file define-
↪metadatablocks.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST -H "Content-
↪type:application/json" --upload-file define-metadatablocks.json "https://demo.
↪dataverse.org/api/dataverses/root/metadatablocks"
```

An alternative to defining metadata blocks at a collection level is to create and use a dataset type. See *Link Dataset Type with Metadata Blocks*.

Determine if a Dataverse Collection Inherits Its Metadata Blocks from Its Parent

Get whether the Dataverse collection is a metadata block root, or does it uses its parent blocks:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/metadatablocks/
↪isRoot"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/metadatablocks/isRoot"
```

Configure a Dataverse Collection to Inherit Its Metadata Blocks from Its Parent

Set whether the Dataverse collection is a metadata block root, or does it uses its parent blocks. Possible values are `true` and `false` (both are valid JSON expressions):

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/dataverses/$ID/
↪metadatablocks/isRoot"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/dataverses/root/metadatablocks/isRoot"
```

> **Note**
>
> Previous endpoints `$SERVER/api/dataverses/$id/metadatablocks/:isRoot` and `POST https://$SERVER/api/dataverses/$id/metadatablocks/:isRoot?key=$apiKey` are deprecated, but supported.

Retrieve a Dataset JSON Schema for a Collection

Retrieves a JSON schema customized for a given collection in order to validate a dataset JSON file prior to creating the dataset:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/datasetSchema"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/datasetSchema"
```

Note: you must have "Add Dataset" permission in the given collection to invoke this endpoint.

While it is recommended to download a copy of the JSON Schema from the collection (as above) to account for any fields that have been marked as required, you can also download a minimal `dataset-schema.json` to get a sense of the schema when no customizations have been made.

Validate Dataset JSON File for a Collection

Validates a dataset JSON file customized for a given collection prior to creating the dataset.

The validation tests for:

- JSON formatting
- required fields
- typeClass must follow these rules:
  - if multiple = true then value must be a list
  - if typeClass = `primitive` the value object is a String or a List of Strings depending on the multiple flag
  - if typeClass = `compound` the value object is a FieldDTO or a List of FieldDTOs depending on the multiple flag
  - if typeClass = `controlledVocabulary` the values are checked against the list of allowed values stored in the database
  - typeName validations (child objects with their required and allowed typeNames are configured automatically by the database schema). Examples include:
    - dsDescription validation includes checks for typeName = `dsDescriptionValue` (required) and `dsDescriptionDate` (optional)
    - datasetContact validation includes checks for typeName = `datasetContactName` (required) and `datasetContactEmail`; `datasetContactAffiliation` (optional)

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/dataverses/$ID/
↪validateDatasetJson" -H 'Content-type:application/json' --upload-file dataset.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/dataverses/root/validateDatasetJson" -H 'Content-type:application/
↪json' --upload-file dataset.json
```

Note: you must have "Add Dataset" permission in the given collection to invoke this endpoint.

List Dataverse Collections to Which a Given Dataset or Dataverse Collection May Be Linked

The user may provide a search term to limit the list of Dataverse Collections returned. The search term will be compared to the name of the Dataverse Collections. The response is a JSON array of the ids, aliases, and names of the Dataverse collections to which a given Dataset or Dataverse Collection may be linked:

For a given Dataverse Collection:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export OBJECT_TYPE=dataverse
export ID=collectionAlias
export SEARCH_TERM=searchOn

curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/dataverses/$ID/$OBJECT_TYPE/
↪linkingDataverses?searchTerm=$SEARCH_TERM"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/dataverses/collectionAlias/dataverse/linkingDataverses?
↪searchTerm=searchOn"
```

For a given Dataset:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export OBJECT_TYPE=dataset
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB
export SEARCH_TERM=searchOn

curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/dataverses/:persistentId/
↪$OBJECT_TYPE/linkingDataverses?searchTerm=SEARCH_TERM&persistentId=$PERSISTENT_
↪IDENTIFIER"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/dataverses/:persistentId/dataset/linkingDataverses?
↪searchTerm=searchOn&persistentId=doi:10.5072/FK2/J8SJZB"
```

You may also add an optional "alreadyLinked=true" parameter to return collections which are already linked to the given Dataset or Dataverse Collection.

List Featured Collections for a Dataverse Collection

The response is a JSON array of the alias strings of the featured collections of a given Dataverse collection identified by `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/dataverses/$ID/featured"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/dataverses/root/featured"
```

Set Featured Collections for a Dataverse Collection

Add featured collections to a given Dataverse collection identified by `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/dataverses/$ID/featured" --
↪upload-file collection-alias.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/dataverses/root/featured" --upload-file collection-alias.json
```

Where collection-alias.json contains a JSON encoded list of collections aliases to be featured (e.g. `["collection1-alias","collection2-alias"]`).

Note: You must have "Edit Dataverse" permission in the given Dataverse to invoke this endpoint. You may only feature collections that are published and owned by or linked to the featuring collection. Also, using this endpoint will only add new featured collections it will not remove collections that have already been featured.

Remove Featured Collections from a Dataverse Collection

Remove featured collections from a given Dataverse collection identified by `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/dataverses/$ID/featured"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/dataverses/root/featured"
```

Note: You must have "Edit Dataverse" permission in the given Dataverse to invoke this endpoint.

Get User Permissions on a Dataverse

This API call returns the permissions that the calling user has on a particular dataverse.

In particular, the user permissions that this API call checks, returned as booleans, are the following:

- Can add a dataverse
- Can add a dataset
- Can view the unpublished dataverse
- Can edit the dataverse
- Can manage the dataverse permissions
- Can publish the dataverse
- Can delete the dataverse

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key: $API_TOKEN" -X GET "$SERVER_URL/api/dataverses/$ID/
↪userPermissions"
```

List Locally FAIR Role Assignees for a Dataverse Collection

Lists the Locally FAIR role assignee identifiers configured for a Dataverse collection identified by `id`. For more about the concept, see *Locally FAIR* in the User Guide.

This API is superuser-only.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/
↪locallyFairRoleAssignees"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/locallyFairRoleAssignees"
```

The response is a JSON array of role assignee identifiers. For example:

```
[
  "@TestUser",
  "&maildomain/harvard.edu"
]
```

Set Locally FAIR Role Assignees for a Dataverse Collection

Replaces the full set locally FAIR role assignee identifiers for a Dataverse collection identified by `id`.

This API is superuser-only.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root
export JSON='["@TestUser","&maildomain/harvard.edu"]'

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT -H "Content-Type: application/json" "$SERVER_
↪URL/api/dataverses/$ID/locallyFairRoleAssignees" -d "$JSON"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT -H "Content-Type:␣
↪application/json" "https://demo.dataverse.org/api/dataverses/root/
↪locallyFairRoleAssignees" -d '["@TestUser","&maildomain/harvard.edu"]'
```

Pass an empty array to clear all locally FAIR role assignees from the collection:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X PUT -H "Content-Type: application/json" "$SERVER_
↪URL/api/dataverses/$ID/locallyFairRoleAssignees" -d '[]'
```

All supplied identifiers must be valid existing role assignee identifiers. Invalid identifiers will result in `400 Bad Request`.

Add a Locally FAIR Role Assignee to a Dataverse Collection

Adds a single locally FAIR role assignee identifier to a Dataverse collection identified by `id`.

This API is superuser-only.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root
export ROLE_ASSIGNEE=&shib/1

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/dataverses/$ID/
↪locallyFairRoleAssignees/$ROLE_ASSIGNEE"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/dataverses/root/locallyFairRoleAssignees/&shib/1"
```

The response includes the updated set of locally FAIR role assignee identifiers.

Delete a Locally FAIR Role Assignee from a Dataverse Collection

Removes a single locally FAIR role assignee identifier from a Dataverse collection identified by `id`.

This API is superuser-only.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root
export ROLE_ASSIGNEE=:authenticated-users

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/dataverses/$ID/
↪locallyFairRoleAssignees/$ROLE_ASSIGNEE"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/dataverses/root/locallyFairRoleAssignees/:authenticated-users"
```

The response includes the updated set of locally FAIR role assignee identifiers. Removing an identifier that is blank or not currently assigned will result in `400 Bad Request`.

Create a Dataset in a Dataverse Collection

A dataset is a container for files as explained in the *Dataset + File Management* section of the User Guide.

To create a dataset, you must supply a JSON file that contains at least the following required metadata fields:

- Title
- Author Name
- Point of Contact Email
- Description Text
- Subject

Submit Incomplete Dataset

Note: This feature requires *dataverse.api.allow-incomplete-metadata* to be enabled and your Solr Schema to be up-to-date with the `datasetValid` field. If not done yet with the version upgrade, you will also need to reindex all dataset after enabling the *dataverse.api.allow-incomplete-metadata* feature.

Providing a `.../datasets?doNotValidate=true` query parameter turns off the validation of metadata. In this situation, only the "Author Name" is required, except for the case when the setting *:MetadataLanguages* is configured and the value of "Dataset Metadata Language" setting of a collection is left with the default "Chosen at Dataset Creation" value. In that case, a language that is a part of the *:MetadataLanguages* list must be declared in the incomplete dataset.

For example, a minimal JSON file, without the language specification, would look like this:

```
{
  "datasetVersion": {
    "metadataBlocks": {
      "citation": {
        "fields": [
```

(continues on next page)

(continued from previous page)

```
          {
            "value": [
              {
                "authorName": {
                  "value": "Finch, Fiona",
                  "typeClass": "primitive",
                  "multiple": false,
                  "typeName": "authorName"
                }
              }
            ],
            "typeClass": "compound",
            "multiple": true,
            "typeName": "author"
          }
        ],
        "displayName": "Citation Metadata"
      }
    }
  }
}
```

The following is an example HTTP call with deactivated validation:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export PARENT=root
export SERVER_URL=https://demo.dataverse.org

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/dataverses/$PARENT/
↪datasets?doNotValidate=true" --upload-file dataset-incomplete.json -H 'Content-
↪type:application/json'
```

Note: You may learn about an instance's support for deposition of incomplete datasets via *Show Support Of Incomplete Metadata Deposition*.

Submit Dataset

As a starting point, you can download `dataset-finch1.json` and modify it to meet your needs. (`dataset-finch1_fr.json` is a variant of this file that includes setting the metadata language (see *:MetadataLanguages*) to French (fr). In addition to this minimal example, you can download `dataset-create-new-all-default-fields.json` which populates all of the metadata fields that ship with a Dataverse installation.)

Note: To set the Template on the new Dataset add "templateId":{$templateId} to the Json payload:

```
{
 "templateId":1234,
 "datasetVersion": {...}
}
```

An invalid templateId will result in a 400 "Error parsing Json: Invalid template id: #".

The curl command below assumes you have kept the name "dataset-finch1.json" and that this file is in your current working directory.

Next you need to figure out the alias or database id of the "parent" Dataverse collection into which you will be creating your new dataset. Out of the box the top level Dataverse collection has an alias of "root" and a database id of "1" but your installation may vary. The easiest way to determine the alias of your root Dataverse collection is to click "Advanced Search" and look at the URL. You may also choose a parent Dataverse collection under the root Dataverse collection.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export PARENT=root
export SERVER_URL=https://demo.dataverse.org

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/dataverses/$PARENT/datasets
↪" --upload-file dataset-finch1.json -H 'Content-type:application/json'
```

The fully expanded example above (without the environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/dataverses/root/datasets" --upload-file "dataset-finch1.json" -H
↪'Content-type:application/json'
```

You should expect an HTTP 200 ("OK") response and JSON indicating the database ID and Persistent ID (PID such as DOI or Handle) that has been assigned to your newly created dataset.

Note

Only a Dataverse installation account with superuser permissions is allowed to include files when creating a dataset via this API. Adding files this way only adds their file metadata to the database, you will need to manually add the physical files to the file system.

Create a Dataset with a Dataset Type (Software, Workflow, Review, etc.)

By default, datasets are given the type "dataset" but if your installation had added additional types (see *Add Dataset Type*), you can specify the type.

Follow *Submit Dataset* as normal but include a line like `"datasetType": "software"` in your JSON. You can check which types are supported by your installation using the *List Dataset Types* API endpoint.

Here is an example JSON file for reference: `dataset-create-software.json`.

See also *Dataset Types*.

Import a Dataset into a Dataverse Collection

> **Note**
>
> This action requires a Dataverse installation account with super-user permissions.

To import a dataset with an existing persistent identifier (PID), the dataset's metadata should be prepared in Dataverse installation's native JSON format. The PID is provided as a parameter at the URL. The following line imports a dataset with the PID `PERSISTENT_IDENTIFIER` to the Dataverse installation, and then releases it:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export DATAVERSE_ID=root
export PERSISTENT_IDENTIFIER=doi:ZZ7/MOSEISLEYDB94

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/dataverses/$DATAVERSE_ID/
↪datasets/:import?pid=$PERSISTENT_IDENTIFIER&release=yes" --upload-file dataset.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/dataverses/root/datasets/:import?pid=doi:ZZ7/MOSEISLEYDB94&
↪release=yes" --upload-file dataset.json
```

The `pid` parameter holds a persistent identifier (such as a DOI or Handle). The import will fail if no PID is provided, or if the provided PID fails validation.

The optional `release` parameter tells the Dataverse installation to immediately publish the dataset. If the parameter is changed to `no`, the imported dataset will remain in `DRAFT` status.

The JSON format is the same as that supported by the native API's *create dataset command*, although it also allows packages. For example:

```
{
  "datasetVersion": {
    "license": {
      "name": "CC0 1.0",
      "uri": "http://creativecommons.org/publicdomain/zero/1.0"
    },
    "protocol":"doi",
    "authority":"10.502",
    "identifier":"ZZ7/MOSEISLEYDB94",
    "metadataBlocks": {
      "citation": {
        "fields": [
          {
            "typeName": "title",
            "multiple": false,
            "value": "Imported dataset with package files No. 3",
            "typeClass": "primitive"
          },
          {
            "typeName": "productionDate",
```

(continues on next page)

(continued from previous page)

```
            "multiple": false,
            "value": "2011-02-23",
            "typeClass": "primitive"
          },
          {
            "typeName": "dsDescription",
            "multiple": true,
            "value": [
              {
                "dsDescriptionValue": {
                  "typeName": "dsDescriptionValue",
                  "multiple": false,
                  "value": "Native Dataset",
                  "typeClass": "primitive"
                }
              }
            ],
            "typeClass": "compound"
          },
          {
            "typeName": "subject",
            "multiple": true,
            "value": [
              "Medicine, Health and Life Sciences"
            ],
            "typeClass": "controlledVocabulary"
          },
          {
            "typeName": "author",
            "multiple": true,
            "value": [
              {
                "authorAffiliation": {
                  "typeName": "authorAffiliation",
                  "multiple": false,
                  "value": "LibraScholar Medical School",
                  "typeClass": "primitive"
                },
                "authorName": {
                  "typeName": "authorName",
                  "multiple": false,
                  "value": "Doc, Bob",
                  "typeClass": "primitive"
                }
              },
              {
                "authorAffiliation": {
                  "typeName": "authorAffiliation",
                  "multiple": false,
                  "value": "LibraScholar Medical School",
                  "typeClass": "primitive"
                },
```

(continues on next page)

(continued from previous page)

```
            "authorName": {
              "typeName": "authorName",
              "multiple": false,
              "value": "Prof, Arthur",
              "typeClass": "primitive"
            }
          }
        ],
        "typeClass": "compound"
      },
      {
        "typeName": "depositor",
        "multiple": false,
        "value": "Prof, Arthur",
        "typeClass": "primitive"
      },
      {
        "typeName": "datasetContact",
        "multiple": true,
        "value": [
          {
            "datasetContactEmail": {
              "typeName": "datasetContactEmail",
              "multiple": false,
              "value": "aprof@mailinator.com",
              "typeClass": "primitive"
            }
          }
        ],
        "typeClass": "compound"
      }
    ],
    "displayName": "Citation Metadata"
  }
},
"files": [
  {
    "description": "",
    "label": "pub",
    "restricted": false,
    "version": 1,
    "datasetVersionId": 1,
    "dataFile": {
      "id": 4,
      "filename": "pub",
      "contentType": "application/vnd.dataverse.file-package",
      "filesize": 1698795873,
      "description": "",
      "storageIdentifier": "162017e5ad5-ee2a2b17fee9",
      "originalFormatLabel": "UNKNOWN",
      "rootDataFileId": -1,
      "checksum": {
```

(continues on next page)

(continued from previous page)

```
            "type": "SHA-1",
            "value": "54bc7ddb096a490474bd8cc90cbed1c96730f350"
          }
        }
      }
    ]
  }
}
```

Before calling the API, make sure the data files referenced by the POSTed JSON are placed in the dataset directory with filenames matching their specified storage identifiers. In installations using POSIX storage, these files must be made readable by the app server user.

Tip

If possible, it's best to avoid spaces and special characters in the storage identifier in order to avoid potential portability problems. The storage identifier corresponds with the filesystem name (or bucket identifier) of the data file, so these characters may cause unpredictability with filesystem tools.

Warning

- This API does not cover staging files (with correct contents, checksums, sizes, etc.) in the corresponding places in the Dataverse installation's filestore.
- This API endpoint does not support importing *files'* persistent identifiers.
- A Dataverse installation can only import datasets with a valid PID that is managed by one of the PID providers that said installation is configured for.

Import a Dataset with a Dataset Type (Software, Workflow, Review, etc.)

By default, datasets are given the type "dataset" but if your installation had added additional types (see *Add Dataset Type*), you can specify the type.

The same native JSON file as above under *Create a Dataset with a Dataset Type (Software, Workflow, Review, etc.)* can be used when importing a dataset.

A file like this is the only difference. Otherwise, follow *Import a Dataset into a Dataverse Collection* as normal.

See also *Dataset Types*.

Import a Dataset into a Dataverse Installation with a DDI file

> **Note**
>
> This action requires a Dataverse installation account with super-user permissions.

To import a dataset with an existing persistent identifier (PID), you have to provide the PID as a parameter at the URL. The following line imports a dataset with the PID `PERSISTENT_IDENTIFIER` to the Dataverse installation, and then releases it:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export DATAVERSE_ID=root
export PERSISTENT_IDENTIFIER=doi:ZZ7/MOSEISLEYDB94

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/dataverses/$DATAVERSE_ID/
↪datasets/:importddi?pid=$PERSISTENT_IDENTIFIER&release=yes" --upload-file ddi_dataset.
↪xml
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/dataverses/root/datasets/:importddi?pid=doi:ZZ7/MOSEISLEYDB94&
↪release=yes" --upload-file ddi_dataset.xml
```

The optional `pid` parameter holds a persistent identifier (such as a DOI or Handle). The import will fail if the provided PID fails validation.

The optional `release` parameter tells the Dataverse installation to immediately publish the dataset. If the parameter is changed to `no`, the imported dataset will remain in `DRAFT` status.

The file is a DDI XML file. A sample DDI XML file may be downloaded here: `ddi_dataset.xml`

Note that DDI XML does not have a field that corresponds to the "Subject" field in Dataverse. Therefore the "Import DDI" API endpoint populates the "Subject" field with `N/A`. To update the "Subject" field one will need to call the *Edit Dataset Metadata* API with a JSON file that contains an update to "Subject" such as `subject-update-metadata.json`. Alternatively, the web interface can be used to add a subject.

> **Warning**
>
> - This API does not handle files related to the DDI file.
> - A Dataverse installation can only import datasets with a valid PID that is managed by one of the PID providers that said installation is configured for.

Publish a Dataverse Collection

In order to publish a Dataverse collection, you must know either its "alias" (which the GUI calls an "identifier") or its database ID.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/dataverses/$ID/actions/
↪:publish"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/dataverses/root/actions/:publish"
```

You should expect a 200 ("OK") response and JSON output.

Retrieve Guestbook Responses for a Dataverse Collection

For more about guestbooks, see *Dataset Guestbooks* in the User Guide.

In order to retrieve the Guestbook Responses for a Dataverse collection, you must know either its "alias" (which the GUI calls an "identifier") or its database ID. If the Dataverse collection has more than one guestbook you may provide the id of a single guestbook as an optional parameter. If no guestbook id is provided the results returned will be the same as pressing the "Download All Responses" button on the Manage Dataset Guestbook page. If the guestbook id is provided then only those responses from that guestbook will be included. The FILENAME parameter is optional, and if it is not included, the responses will be displayed in the console.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root
export GUESTBOOK_ID=1
export FILENAME=myResponses.csv

curl -H  "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/guestbookResponses?
↪guestbookId=$GUESTBOOK_ID" -o $FILENAME
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/guestbookResponses?guestbookId=1" -o myResponses.csv
```

Guestbooks

Create a Guestbook for a Dataverse Collection

For more about guestbooks, see *Dataset Guestbooks* in the User Guide.

Create a Guestbook that can be selected for a Dataset. You must have "EditDataverse" permission on the Dataverse collection.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root
export JSON='{"name": "my test guestbook","enabled": true,"emailRequired": true,
↪"nameRequired": true,"institutionRequired": false,"positionRequired": false,
↪"customQuestions": [{"question": "how is your day","required": true,"displayOrder": 0,
↪"type": "text","hidden": false}]}'

curl -POST -H  "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/guestbooks/{ID}" -d "$JSON"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -POST -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.
↪dataverse.org/api/guestbooks/root" -d '{"name": "my test guestbook","enabled": true,
↪"emailRequired": true,"nameRequired": true,"institutionRequired": false,
↪"positionRequired": false}'
```

Get a list of Guestbooks for a Dataverse Collection

For more about guestbooks, see *Dataset Guestbooks* in the User Guide.

Get a list of Guestbooks for a Dataverse Collection

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/guestbooks/{ID}/list"`
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/guestbooks/root/list"
```

To include the inherited Guestbooks of the Dataverse Collection add the query parameter *includeInherited=true*

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/guestbooks/root/list?includeInherited=true"
```

Note

By adding the query param "includeStats=true" *usageCount* and *responseCount* values can be added to the response.

Get a Guestbook for a Dataverse Collection

For more about guestbooks, see *Dataset Guestbooks* in the User Guide.

Get a Guestbook by its id

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1234

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/guestbooks/{ID}"`
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/guestbooks/1234"
```

Update a Guestbook for a Dataverse Collection

For more about guestbooks, see *Dataset Guestbooks* in the User Guide.

Update a Guestbook that can be selected for a Dataset. You must have "EditDataverse" permission on the Dataverse collection.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1234

curl -PUT -H  "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/guestbooks/{ID}" -d "$JSON"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -PUT -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.
↪dataverse.org/api/guestbooks/1234" -d "$JSON"
```

Enable or Disable a Guestbook for a Dataverse Collection

For more about guestbooks, see *Dataset Guestbooks* in the User Guide.

Use this endpoint to enable or disable the Guestbook. A Guestbook can not be deleted or modified since there may be responses linked to it. You must have "EditDataverse" permission on the Dataverse collection.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export dataverseIdentifier=root
export ID=1234

curl -X PUT -d 'true' -H  "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/guestbooks/
↪{dataverseIdentifier}/{ID}/enabled"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -X PUT -d 'true' -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://
↪demo.dataverse.org/api/guestbooks/root/1234"
```

Retrieve Guestbook Responses for a Guestbook

For more about guestbooks, see *Dataset Guestbooks* in the User Guide.

In order to retrieve the Guestbook Responses for a Guestbook within a Dataverse collection, you must know the ID if the Guestbook. This API also supports pagination by passing a page limit and an optional offset (starting point). The resulting Json will include 'Next' and 'Prev' urls for navigation as well as the total number of responses. The resulting Json will be more detailed than that of the *Retrieve Guestbook Responses for a Dataverse Collection* CSV response file by including Guestbook metadata as well as Guestbook Response metadata.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -H  "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/guestbooks/$ID/responses"
curl -H  "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/guestbooks/$ID/responses?limit10&
↪offset=0"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/guestbooks/1/responses"
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/guestbooks/1/responses?limit10&offset=0"
```

Change Collection Attributes

```
curl -X PUT -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/attribute/
↪$ATTRIBUTE?value=$VALUE"
```

The following attributes are supported:

- `alias` Collection alias
- `name` Name
- `description` Description
- `affiliation` Affiliation
- `filePIDsEnabled` ("true" or "false") Restricted to use by superusers and only when the *:AllowEnablingFilePIDsPerCollection* setting is true. Enables or disables registration of file-level PIDs in datasets within the collection (overriding the instance-wide setting).

- `requireFilesToPublishDataset` ("true" or "false") Restricted to use by superusers. Defines if Dataset version needs files in order to be published or submitted for review. If not set the determination will be made through inheritance by checking the owners of this collection. Publishing by a superusers will not be blocked.
- `allowedDatasetTypes` Restricted to use by superusers. By default "dataset" is implied. Pass a comma-separated list of dataset types (e.g. "dataset,software"). You cannot unset this attribute so if you want to delete a dataset type, set `allowedDatasetTypes` to a dataset type you won't be deleting. See also *Dataset Types*.

See also *Update a Dataverse Collection*.

Update Collection Input Levels

Updates the dataset field type input levels in a collection.

Please note that this endpoint does not change previously updated input levels of the collection page, so if you want to add new levels or modify existing ones, you will need to include them in the JSON request body. In order to delete input levels you must call this API with an empty list to delete all of the input levels, then call this API with the new list of input levels.

If one of the input levels corresponds to a dataset field type belonging to a metadata block that does not exist in the collection, the metadata block will be added to the collection.

This endpoint expects a JSON with the following format:

```
[
  {
    "datasetFieldTypeName": "datasetFieldTypeName1",
    "required": true,
    "include": true,
    "displayOnCreate": null
  },
  {
    "datasetFieldTypeName": "datasetFieldTypeName2",
    "required": true,
    "include": true,
    "displayOnCreate": true
  }
]
```

Note

Required fields will always be displayed regardless of their displayOnCreate setting, as this is necessary for dataset creation. When displayOnCreate is null, the field's default display behavior is used.

Parameters:

- `datasetFieldTypeName`: Name of the metadata field
- `required`: Whether the field is required (boolean)
- `include`: Whether the field is included (boolean)
- `displayOnCreate` (optional): Whether the field is displayed during dataset creation, even when not required (boolean)

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root
export JSON='[{"datasetFieldTypeName":"geographicCoverage", "required":true, "include
↪":true}, {"datasetFieldTypeName":"country", "required":true, "include":true}]'

curl -X PUT -H "X-Dataverse-key: $API_TOKEN" -H "Content-Type:application/json" "$SERVER_
↪URL/api/dataverses/$ID/inputLevels" -d "$JSON"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -X PUT -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Content-
↪Type:application/json" "https://demo.dataverse.org/api/dataverses/root/inputLevels" -d
↪'[{"datasetFieldTypeName":"geographicCoverage", "required":true, "include":false}, {
↪"datasetFieldTypeName":"country", "required":true, "include":false}]'
```

Collection Storage Quotas

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/storage/quota"
```

Will output the storage quota allocated (in bytes), or a message indicating that the quota is not defined for the collection. If this is an unpublished collection, the user must have the `ViewUnpublishedDataverse` permission. With an optional query parameter `showInherited=true` it will show the applicable quota potentially defined on the nearest parent when the collection does not have a quota configured directly.

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/storage/use"
```

Will output the dynamically cached total storage size (in bytes) used by the collection. The user identified by the API token must have the `Edit` permission on the collection.

To set or change the storage allocation quota for a collection:

```
curl -X PUT -H "X-Dataverse-key:$API_TOKEN" -d $SIZE_IN_BYTES "$SERVER_URL/api/
↪dataverses/$ID/storage/quota"
```

This API is superuser-only.

To delete a storage quota configured for a collection:

```
curl -X DELETE -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/storage/
↪quota"
```

This API is superuser-only.

Storage Quotas on Individual Datasets

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/datasets/$ID/storage/quota"
```

Will output the storage quota allocated (in bytes), or a message indicating that the quota is not defined for this dataset. If this is an unpublished dataset, the user must have the `ViewUnpublishedDataset` permission. With an optional query parameter `showInherited=true` it will show the applicable quota potentially defined on the nearest parent collection when the dataset does not have a quota configured directly.

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/datasets/$ID/storage/use"
```

Will output the dynamically cached total storage size (in bytes) used by the dataset. The user identified by the API token must have the `Edit` permission on the dataset.

To set or change the storage allocation quota for a dataset:

```
curl -X PUT -H "X-Dataverse-key:$API_TOKEN" -d $SIZE_IN_BYTES "$SERVER_URL/api/datasets/
↪$ID/storage/quota"
```

This API is superuser-only.

To delete a storage quota configured for a dataset:

```
curl -X DELETE -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/datasets/$ID/storage/
↪quota"
```

This API is superuser-only.

The following convenience API shows the dynamic values of the *remaining* storage size and/or file number quotas on the dataset, if present. For example:

```
curl  -H "X-Dataverse-key: $API_TOKEN" "http://localhost:8080/api/datasets/$dataset-id/
↪uploadlimits"
{
  "status": "OK",
  "data": {
    "uploadLimits": {
      "numberOfFilesRemaining": 20,
      "storageQuotaRemaining": 1048576
    }
  }
}
```

Or, when neither limit is present:

```
{
  "status": "OK",
  "data": {
    "uploadLimits": {}
  }
}
```

This API requires the Edit permission on the dataset.

Use the `/settings` API to enable or disable the enforcement of storage quotas that are defined across the instance via the following setting:

```
curl -X PUT -d 'true' http://localhost:8080/api/admin/settings/:UseStorageQuotas
```

List All Collection Featured Items

List the featured items configured for a given Dataverse collection `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/featuredItems"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/dataverses/root/featuredItems"
```

Update All Collection Featured Items

Updates all featured items in the given Dataverse collection `id`.

The data sent to the endpoint represents the desired final state of the featured items in the Dataverse collection and overwrites any existing featured items configuration.

The parameters `id`, `content`, `displayOrder`, and `fileName` must be specified as many times as the number of items we want to add or update. The order in which these parameters are repeated must match to ensure they correspond to the same featured item.

The `file` parameter must be specified for each image we want to attach to featured items. Note that images can be shared between featured items, so `fileName` can have the same value in different featured items.

The `id` parameter must be `0` for new items or set to the item's identifier for updates. The `fileName` parameter should be empty to exclude an image or match the name of a file sent in a `file` parameter to set a new image. `keepFile` must always be set to `false`, unless it's an update to a featured item where we want to preserve the existing image, if one exists.

The `type` and `dvObjectIdentifier` parameters are optional. These allow you to link the featured item to a Dataverse, Dataset, or Datafile. The `dvObjectIdentifier` can be passed as the alias, id, or the persistent identifier and the `type` must be passed as either "dataverse", "dataset", or "datafile", depending on the type of object. If no `dvObjectIdentifier` is passed the `type` will default to "custom" designating no linked object.

Note that any existing featured item not included in the call with its associated identifier and corresponding properties will be removed from the collection.

The following example creates two featured items, with an image and a dataset assigned to the second one:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

export FIRST_ITEM_CONTENT='Content 1'
export FIRST_ITEM_DISPLAY_ORDER=1

export SECOND_ITEM_IMAGE_FILENAME='image.png'
```

(continues on next page)

(continued from previous page)

```
export SECOND_ITEM_CONTENT='Content 2'
export SECOND_ITEM_DISPLAY_ORDER=2
export SECOND_ITEM_TYPE="dataset"
export SECOND_ITEM_DVOBJECT="doi:ZZ7/MOSEISLEYDB94"

curl -H "X-Dataverse-key:$API_TOKEN" \
     -X PUT \
     -F "id=0" -F "id=0" \
     -F "content=$FIRST_ITEM_CONTENT" -F "content=$SECOND_ITEM_CONTENT" \
     -F "displayOrder=$FIRST_ITEM_DISPLAY_ORDER" -F "displayOrder=$SECOND_ITEM_DISPLAY_
↪ORDER" \
     -F "fileName=" -F "fileName=$SECOND_ITEM_IMAGE_FILENAME" \
     -F "keepFile=false" -F "keepFile=false" \
     -F "file=@$SECOND_ITEM_IMAGE_FILENAME" \
     -F "type=" -F "type=@$SECOND_ITEM_TYPE" \
     -F "dvObjectIdentifier=" -F "dvObjectIdentifier=@$SECOND_ITEM_DVOBJECT" \
     "$SERVER_URL/api/dataverses/$ID/featuredItems"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" \
     -X PUT \
     -F "id=0" -F "id=0" \
     -F "content=Content 1" -F "content=Content 2" \
     -F "displayOrder=1" -F "displayOrder=2" \
     -F "fileName=" -F "fileName=image.png" \
     -F "keepFile=false" -F "keepFile=false" \
     -F "file=@image.png" \
     -F "type=" -F "type=dataset" \
     -F "dvObjectIdentifier=" -F "dvObjectIdentifier=doi:ZZ7/MOSEISLEYDB94" \
     "https://demo.dataverse.org/api/dataverses/root/featuredItems"
```

The following example creates one featured item and updates a second one, keeping the existing image it may have had but removes the dataset link and defaults the type to "custom":

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" \
     -X PUT \
     -F "id=0" -F "id=1" \
     -F "content=Content 1" -F "content=Updated content 2" \
     -F "displayOrder=1" -F "displayOrder=2" \
     -F "fileName=" -F "fileName=" \
     -F "keepFile=false" -F "keepFile=true" \
     "https://demo.dataverse.org/api/dataverses/root/featuredItems"
```

Delete All Collection Featured Items

Deletes the featured items configured for a given Dataverse collection `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE "$SERVER_URL/api/dataverses/$ID/
↪featuredItems"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/dataverses/root/featuredItems"
```

Create a Collection Featured Item

Creates a featured item in the given Dataverse collection `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export IMAGE_FILENAME='image.png'
export CONTENT='Content for featured item.'
export DISPLAY_ORDER=1
export SERVER_URL=https://demo.dataverse.org
export ID=root

curl -H "X-Dataverse-key:$API_TOKEN" -X POST -F "file=@$IMAGE_FILENAME" -F "content=
↪$CONTENT" -F "displayOrder=$DISPLAY_ORDER" "$SERVER_URL/api/dataverses/$ID/
↪featuredItems"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST -F "file=@image.
↪png" -F "content=Content for featured item." -F "displayOrder=1" "https://demo.
↪dataverse.org/api/dataverses/root/featuredItems"
```

A featured item may or may not contain an image. If you wish to create it without an image, omit the file parameter in the request.

Update a Collection Featured Item

Updates a featured item given its `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export IMAGE_FILENAME='image.png'
export CONTENT='Content for featured item.'
export DISPLAY_ORDER=1
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT -F "file=@$IMAGE_FILENAME" -F "content=
↪$CONTENT" -F "displayOrder=$DISPLAY_ORDER" "$SERVER_URL/api/dataverseFeaturedItems/$ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT -F "file=@image.png
↪" -F "content=Content for featured item." -F "displayOrder=1" "https://demo.dataverse.
↪org/api/dataverseFeaturedItems/1"
```

`content` and `displayOrder` must always be provided; otherwise, an error will occur. Use the `file` parameter to set a new image for the featured item. To keep the existing image, omit `file` and send `keepFile=true`. To remove the image, omit the file parameter.

Updating the featured item keeping the existing image:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT -F "keepFile=true"␣
↪-F "content=Content for featured item." -F "displayOrder=1" "https://demo.dataverse.
↪org/api/dataverseFeaturedItems/1"
```

Updating the featured item removing the existing image:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT -F
↪"content=Content for featured item." -F "displayOrder=1" "https://demo.dataverse.org/
↪api/dataverseFeaturedItems/1"
```

Delete a Collection Featured Item

Deletes a featured item given its `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/dataverseFeaturedItems/
↪$ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/dataverseFeaturedItems/1"
```

Get a Collection Featured Item Image

Returns the image of a featured item if one is assigned, given the featured item `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/access/
↪dataverseFeaturedItemImage/{ID}"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/access/dataverseFeaturedItemImage/1"
```

List Templates of a Collection

Lists the templates for a given Dataverse collection `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/dataverses/{ID}/templates"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/dataverses/1/templates"
```

List Single Template by its Identifier

Gets the json representation of a template by its `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/dataverses/{ID}/template"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/dataverses/1/template"
```

You must have Create Dataset permission within the given dataverse collection to invoke this api.

Delete a Template by its Identifier

Deletes a template by its `id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/dataverses/{ID}/template"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/dataverses/1/template"
```

You must have Edit Dataverse permission within the given dataverse collection to invoke this api.

Create a Template for a Collection

Creates a template for a given Dataverse collection `id`.

To create the template, you must send a JSON file. Your JSON file might look like `dataverse-template.json` which you would send to the Dataverse installation like this:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -H "X-Dataverse-key: $API_TOKEN" -X POST "$SERVER_URL/api/dataverses/{ID}/templates
↪" --upload-file dataverse-template.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/dataverses/1/templates" --upload-file dataverse-template.json
```

Update the Metadata and Instructions of a Template

Updates the metadata and instructions of a template with a given `id`.

To update the template, you must send a JSON file. Your JSON file might look like `template-update-metadata.json` which you would send to the Dataverse installation like this:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -H "X-Dataverse-key: $API_TOKEN" -X PUT "$SERVER_URL/api/dataverses/{ID}/metadata" -
↪-upload-file template-update-metadata.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/dataverses/1/metadata" --upload-file template-update-metadata.json
```

Update the License or Terms Of Use of a Template

Updates the license or custom terms of use of a template with a given `id`.

To update the template, you must send a JSON file containing either the name of an active license or custom terms of use. Your JSON file might look like `template-update-license.json` or `template-update-terms.json` which you would send to the Dataverse installation like this:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -H "X-Dataverse-key: $API_TOKEN" -X PUT "$SERVER_URL/api/dataverses/{ID}/
↪licenseTerms" --upload-file template-update-license.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/dataverses/1/licenseTerms" --upload-file template-update-license.json
```

Update the Terms Of Access of a Template

Updates the terms of access of a template with a given `id`.

To update the template, you must send a JSON file containing either the name of an active license or custom terms of use. Your JSON file might look like `template-update-access.json` which you would send to the Dataverse installation like this:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -H "X-Dataverse-key: $API_TOKEN" -X PUT "$SERVER_URL/api/dataverses/{ID}/access" --
↪upload-file template-update-access.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/dataverses/1/access" --upload-file template-update-access.json
```

Set a Default Template for a Collection

Sets a template as the default template for a collection:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1
export TEMPLATEID=2

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/dataverses/$ID/template/
↪default/$TEMPLATEID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/dataverses/1/template/default/2"
```

You must have Edit Dataverse permission within the given dataverse collection to invoke this api.

Remove a Default Template for a Collection

Remove a template as the default template for a collection(Note: the template is not deleted; it will still be available for use in the collection):

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1


curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/dataverses/$ID/template/
↪default"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/dataverses/1/template/default"
```

You must have Edit Dataverse permission within the given dataverse collection to invoke this api.

Dataverse Role Assignment History

Get the history of role assignments for a collection. This API call returns a list of role assignments and revocations for the specified dataset.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -H "X-Dataverse-key:$API_TOKEN" -H "Accept: application/json" "$SERVER_URL/api/
↪dataverses/$ID/assignments/history"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Accept: application/
↪json" "https://demo.dataverse.org/api/dataverses/3/assignments/history"
```

You can also use the collection alias instead of the numeric id:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export DV_ALIAS=dvAlias

curl -H "X-Dataverse-key:$API_TOKEN" -H "Accept: application/json" "$SERVER_URL/api/
↪dataverses/$DV_ALIAS/assignments/history"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Accept: application/
↪json" "https://demo.dataverse.org/api/datasets/dvAlias/assignments/history"
```

The response is a JSON array of role assignment history entries with the following structure for each entry:

```
{
  "definedOn": "1",
  "assigneeIdentifier": "@user1",
  "roleName": "Admin",
  "assignedBy": "@dataverseAdmin",
  "assignedAt": "2023-01-01T12:00:00Z",
  "revokedBy": null,
  "revokedAt": null
}
```

For revoked assignments, the "revokedBy" and "revokedAt" fields will contain values instead of null.

To retrieve the history in CSV format, change the Accept header to "text/csv":

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=3

curl -H "X-Dataverse-key:$API_TOKEN" -H "Accept: text/csv" "$SERVER_URL/api/dataverses/
↪$ID/assignments/history"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Accept: text/csv"
↪"https://demo.dataverse.org/api/dataverses/3/assignments/history"
```

The CSV response has column headers mirroring the JSON entries. They are internationalized (when internationalization is configured).

Note: This feature requires the *dataverse.feature.role-assignment-history* feature flag to be enabled.

5.6.2 Datasets

Note Creation of new datasets is done with a `POST` onto a Dataverse collection. See the Dataverse Collections section above.

Dataset Version Specifiers

In all commands below, dataset versions can be referred to as:

- `:draft` the draft version, if any
- `:latest` either a draft (if exists) or the latest published version.
- `:latest-published` the latest published version
- `x.y` a specific version, where `x` is the major version number and `y` is the minor version number.
- `x` same as `x.0`

Get JSON Representation of a Dataset

> **Note**
>
> Datasets can be accessed using persistent identifiers. This is done by passing the constant `:persistentId` where the numeric id of the dataset is expected, and then passing the actual persistent id as a query parameter with the name `persistentId`.

If a user with EditDataset permissions wants to ignore the setting `ExcludeEmailFromExport` in order to see the contact email, they must include the `ignoreSettingExcludeEmailFromExport` query parameter (Required by SPA).

Example: Getting the dataset whose DOI is *10.5072/FK2/J8SJZB*:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/datasets/:persistentId/?
↪persistentId=$PERSISTENT_IDENTIFIER&ignoreSettingExcludeEmailFromExport"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:$API_TOKEN" "https://demo.dataverse.org/api/datasets/
↪:persistentId/?persistentId=doi:10.5072/FK2/J8SJZB&ignoreSettingExcludeEmailFromExport"
```

Getting its draft version:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB

curl -H "X-Dataverse-key:$API_TOKEN" "https://$SERVER/api/datasets/:persistentId/
↪versions/:draft?persistentId=$PERSISTENT_IDENTIFIER"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:$API_TOKEN" "https://demo.dataverse.org/api/datasets/
↪:persistentId/versions/:draft?persistentId=doi:10.5072/FK2/J8SJZB"
```

Show the dataset whose database id is passed:

```
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl "$SERVER_URL/api/datasets/$ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/24"
```

The dataset id can be extracted from the response retrieved from the API which uses the persistent identifier (`/api/datasets/:persistentId/?persistentId=$PERSISTENT_IDENTIFIER`).

If you want to include the Dataverse collections that this dataset is part of, you must set `returnOwners` query parameter to `true`.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24?returnOwners=true"
```

List Versions of a Dataset

List versions of the dataset:

```
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl "$SERVER_URL/api/datasets/$ID/versions"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/24/versions"
```

It returns a list of versions with their metadata, and file list:

```
{
  "status": "OK",
  "data": [
    {
      "id": 7,
      "datasetId": 24,
      "datasetPersistentId": "doi:10.5072/FK2/U6AEZM",
      "storageIdentifier": "file://10.5072/FK2/U6AEZM",
      "versionNumber": 2,
      "versionMinorNumber": 0,
      "versionState": "RELEASED",
      "lastUpdateTime": "2015-04-20T09:58:35Z",
      "releaseTime": "2015-04-20T09:58:35Z",
      "createTime": "2015-04-20T09:57:32Z",
      "license": {
        "name": "CC0 1.0",
        "uri": "http://creativecommons.org/publicdomain/zero/1.0",
        "iconUri": "https://licensebuttons.net/p/zero/1.0/88x31.png",
        "rightsIdentifier": "CC0",
        "rightsIdentifierScheme": "Creative Commons",
        "schemeUri": "https://creativecommons.org/",
        "languageCode": "en",
      },
      "termsOfAccess": "You need to request for access.",
      "fileAccessRequest": true,
      "metadataBlocks": {...},
      "files": [...]
    },
    {
      "id": 6,
      "datasetId": 24,
      "datasetPersistentId": "doi:10.5072/FK2/U6AEZM",
      "storageIdentifier": "file://10.5072/FK2/U6AEZM",
      "versionNumber": 1,
      "versionMinorNumber": 0,
```

(continues on next page)

(continued from previous page)

```
      "versionState": "RELEASED",
      "UNF": "UNF:6:y4dtFxWhBaPM9K/jlPPuqg==",
      "lastUpdateTime": "2015-04-20T09:56:34Z",
      "releaseTime": "2015-04-20T09:56:34Z",
      "createTime": "2015-04-20T09:43:45Z",
      "license": {
        "name": "CC0 1.0",
        "uri": "http://creativecommons.org/publicdomain/zero/1.0",
        "iconUri": "https://licensebuttons.net/p/zero/1.0/88x31.png",
        "rightsIdentifier": "CC0",
        "rightsIdentifierScheme": "Creative Commons",
        "schemeUri": "https://creativecommons.org/",
        "languageCode": "en",
      },
      "termsOfAccess": "You need to request for access.",
      "fileAccessRequest": true,
      "metadataBlocks": {...},
      "files": [...]
    }
  ]
}
```

The optional `excludeFiles` parameter specifies whether the files should be listed in the output. It defaults to `true`, preserving backward compatibility. (Note that for a dataset with a large number of versions and/or files having the files included can dramatically increase the volume of the output). A separate `/files` API can be used for listing the files, or a subset thereof in a given version.

The optional `excludeMetadataBlocks` parameter specifies whether the metadata blocks should be listed in the output. It defaults to `false`, preserving backward compatibility. (Note that for a dataset with a large number of versions and/or metadata blocks having the metadata blocks included can dramatically increase the volume of the output).

The optional `offset` and `limit` parameters can be used to specify the range of the versions list to be shown. This can be used to paginate through the list in a dataset with a large number of versions.

Get Version of a Dataset

Show a version of the dataset. The output includes any metadata blocks the dataset might have:

```
export SERVER_URL=https://demo.dataverse.org
export ID=24
export VERSION=1.0

curl "$SERVER_URL/api/datasets/$ID/versions/$VERSION?excludeFiles=false"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0?excludeFiles=false"
```

Note: the data object will contain the fields `datasetId`, `datasetPersistentId` (since 4.18) and `datasetType` (since 6.7). All of these fields are equal and constant for all versions of a dataset.

The optional `excludeFiles` parameter specifies whether the files should be listed in the output (defaults to `true`). Note that a separate `/files` API can be used for listing the files, or a subset thereof in a given version.

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0?excludeMetadataBlocks=false
↪"
```

The optional `excludeMetadataBlocks` parameter specifies whether the metadata blocks should be listed in the output (defaults to `false`).

By default, deaccessioned dataset versions are not included in the search when applying the :latest or :latest-published identifiers. Additionally, when filtering by a specific version tag, you will get a "not found" error if the version is deaccessioned and you do not enable the `includeDeaccessioned` option described below.

If you want to include deaccessioned dataset versions, you must set `includeDeaccessioned` query parameter to `true`.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0?includeDeaccessioned=true"
```

If you want to include the Dataverse collections that this dataset version is part of, you must set `returnOwners` query parameter to `true`.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0?returnOwners=true"
```

Export Metadata of a Dataset in Various Formats

Export the metadata of either the current published version or the draft version of a dataset in various formats.

To get a list of available formats, see *Available Dataset Metadata Exporters* and *Get Dataset Metadata Export Formats*.

If you don't specify a version (see *Dataset Version Specifiers*), `:latest-published` is assumed and an API token is not necessary. `:draft` is supported if you pass an API token that has access. If you try to pass a version number (e.g. "1.0"), it will only work if it happens to be the latest published version. That is to say, for published versions, only the latest published version is supported.

See also *Batch Exports Through the API* and the note below:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB
export METADATA_FORMAT=ddi
export VERSION=:draft
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

curl -H "X-Dataverse-key: $API_TOKEN" "$SERVER_URL/api/datasets/export?exporter=
↪$METADATA_FORMAT&persistentId=$PERSISTENT_IDENTIFIER&version=$VERSION"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/datasets/export?exporter=ddi&persistentId=doi:10.5072/FK2/J8SJZB&version=:draft
↪"
```

Available Dataset Metadata Exporters

The following dataset metadata exporters ship with Dataverse:

- `croissant`
- `croissantSlim`
- `Datacite`
- `dataverse_json`
- `dcterms`
- `ddi`
- `oai_datacite`
- `oai_dc`
- `oai_ddi`
- `OAI_ORE`
- `schema.org`

These are the strings to pass as `$METADATA_FORMAT` in the examples above. Descriptive names for each format can be found under *Supported Metadata Export Formats* in the User Guide.

Additional exporters can be enabled, as described under *External Metadata Exporters* in the Installation Guide. The machine-readable name/identifier for each external exporter can be found under *Inventory of External Exporters*. If you are interested in creating your own exporter, see *Metadata Export Formats*.

To discover the machine-readable name of exporters (e.g. `ddi`) that have been enabled on the installation of Dataverse you are using see *Get Dataset Metadata Export Formats*. Alternatively, you can use the Signposting "linkset" API documented under *Retrieve Signposting Information*.

To discover the machine-readable name of exporters generally, check *Inventory of External Exporters* or `getFormatName` in the exporter's source code.

Schema.org JSON-LD

Please note that the `schema.org` format has changed in backwards-incompatible ways after Dataverse 4.9.4:

- "description" was a single string and now it is an array of strings.
- "citation" was an array of strings and now it is an array of objects.

Both forms are valid according to Google's Structured Data Testing Tool at https://search.google.com/structured-data/testing-tool . Schema.org JSON-LD is an evolving standard that permits a great deal of flexibility. For example, https://schema.org/docs/gs.html#schemaorg_expected indicates that even when objects are expected, it's ok to just use text. As with all metadata export formats, we will try to keep the Schema.org JSON-LD format backward-compatible to make integrations more stable, despite the flexibility that's afforded by the standard.

The standard has further evolved into a format called Croissant. For details, see *Croissant Metadata in the <head> of Dataset Landing Pages* in the Admin Guide.

The `schema.org` format changed after Dataverse 6.4 as well. Previously its content type was "application/json" but now it is "application/ld+json".

List Files in a Dataset

Lists all the file metadata, for the given dataset and version:

```
export SERVER_URL=https://demo.dataverse.org
export ID=24
export VERSION=1.0

curl "$SERVER_URL/api/datasets/$ID/versions/$VERSION/files"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files"
```

This endpoint supports optional pagination, through the `limit` and `offset` query parameters.

To aid in pagination the JSON response also includes the total number of rows (totalCount) available.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files?limit=10&offset=20"
```

Category name filtering is also optionally supported. To return files to which the requested category has been added.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files?categoryName=Data"
```

Tabular tag name filtering is also optionally supported. To return files to which the requested tabular tag has been added.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files?tabularTagName=Survey
↪"
```

Content type filtering is also optionally supported. To return files matching the requested content type.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files?contentType=image/png
↪"
```

Filtering by search text is also optionally supported. The search will be applied to the labels and descriptions of the dataset files, to return the files that contain the text searched in one of such fields.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files?searchText=word"
```

File access filtering is also optionally supported. In particular, by the following possible values:

- `Public`
- `Restricted`
- `EmbargoedThenRestricted`
- `EmbargoedThenPublic`
- `RetentionPeriodExpired`

If no filter is specified, the files will match all of the above categories.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files?accessStatus=Public"
```

Ordering criteria for sorting the results is also optionally supported. In particular, by the following possible values:

- `NameAZ` (Default)
- `NameZA`
- `Newest`
- `Oldest`
- `Size`
- `Type`

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files?orderCriteria=Newest"
```

Please note that both filtering and ordering criteria values are case sensitive and must be correctly typed for the endpoint to recognize them.

By default, deaccessioned dataset versions are not included in the search when applying the :latest or :latest-published identifiers. Additionally, when filtering by a specific version tag, you will get a "not found" error if the version is deaccessioned and you do not enable the `includeDeaccessioned` option described below.

If you want to include deaccessioned dataset versions, you must set `includeDeaccessioned` query parameter to `true`.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files?
↪includeDeaccessioned=true"
```

Note

Keep in mind that you can combine all of the above query parameters depending on the results you are looking for.

Get File Counts in a Dataset

Get file counts, for the given dataset and version.

The returned file counts are based on different criteria:

- Total (The total file count)
- Per content type
- Per category name
- Per tabular tag name
- Per access status (Possible values: Public, Restricted, EmbargoedThenRestricted, EmbargoedThenPublic, RetentionPeriodExpired)

```
export SERVER_URL=https://demo.dataverse.org
export ID=24
export VERSION=1.0

curl "$SERVER_URL/api/datasets/$ID/versions/$VERSION/files/counts"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files/counts"
```

Category name filtering is optionally supported. To return counts only for files to which the requested category has been added.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files/counts?
↪categoryName=Data"
```

Tabular tag name filtering is also optionally supported. To return counts only for files to which the requested tabular tag has been added.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files/counts?
↪tabularTagName=Survey"
```

Content type filtering is also optionally supported. To return counts only for files matching the requested content type.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files/counts?
↪contentType=image/png"
```

Filtering by search text is also optionally supported. The search will be applied to the labels and descriptions of the dataset files, to return counts only for files that contain the text searched in one of such fields.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files/counts?
↪searchText=word"
```

File access filtering is also optionally supported. In particular, by the following possible values:

- `Public`
- `Restricted`
- `EmbargoedThenRestricted`
- `EmbargoedThenPublic`
- `RetentionPeriodExpired`

If no filter is specified, the files will match all of the above categories.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files/counts?
↪accessStatus=Public"
```

By default, deaccessioned dataset versions are not supported by this endpoint and will be ignored in the search when applying the :latest or :latest-published identifiers. Additionally, when filtering by a specific version tag, you will get a not found error if the version is deaccessioned and you do not enable the option described below.

If you want to include deaccessioned dataset versions, you must specify this through the `includeDeaccessioned` query parameter.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/files/counts?
↪includeDeaccessioned=true"
```

Please note that filtering values are case sensitive and must be correctly typed for the endpoint to recognize them.

Keep in mind that you can combine all of the above query parameters depending on the results you are looking for.

View Dataset Files and Folders as a Directory Index

Provides a *crawlable* view of files and folders within the given dataset and version:

```
curl "$SERVER_URL/api/datasets/${ID}/dirindex/"
# or
curl "${SERVER_URL}/api/datasets/:persistentId/dirindex?persistentId=doi:${PERSISTENT_ID}
↪"
```

Optional parameters:

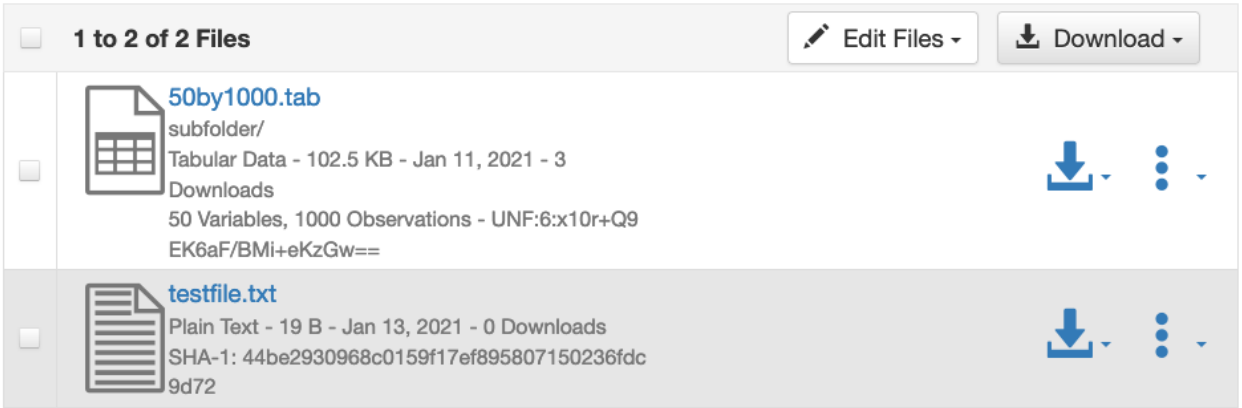
- `folder` - A subfolder within the dataset (default: top-level view of the dataset)
- `version` - Specifies the version (default: latest published version)
- `original=true` - Download original versions of ingested tabular files.

This API outputs a simple html listing, based on the standard Apache directory index, with Access API download links for individual files, and recursive calls to the API above for sub-folders.

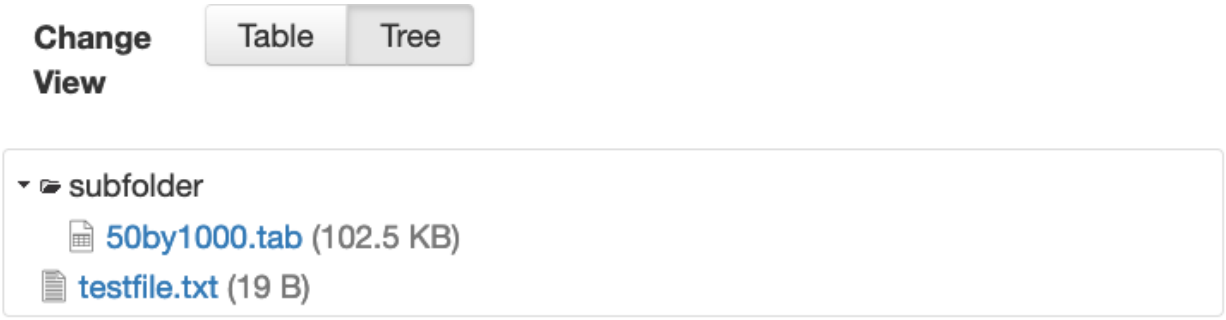
Using this API, `wget --recursive` (or a similar crawling client) can be used to download all the files in a dataset, preserving the file names and folder structure; without having to use the download-as-zip API. In addition to being faster (zipping is a relatively resource-intensive operation on the server side), this process can be restarted if interrupted (with `wget --continue` or equivalent) - unlike zipped multi-file downloads that always have to start from the beginning.

On a system that uses S3 with download redirects, the individual file downloads will be handled by S3 directly, without having to be proxied through the Dataverse application.

For example, if you have a dataset version with 2 files, one with the folder named "subfolder":



or, as viewed as a tree on the dataset page:



The output of the API for the top-level folder (`/api/datasets/{dataset}/dirindex/`) will be as follows:

Index of folder / in dataset doi:XXX/YY/ZZZZ (v. MM)

Name	Last Modified	Size	Description
subfolder/		-	-
testfile.txt	13-January-2021 22:35	19 B	

with the underlying html source:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html><head><title>Index of folder /</title></head>
<body><h1>Index of folder / in dataset doi:XXX/YY/ZZZZ (v. MM)</h1>
<table>
<tr><th>Name</th><th>Last Modified</th><th>Size</th><th>Description</th></tr>
<tr><th colspan="4"><hr></th></tr>
<tr><td><a href="/api/datasets/NNNN/dirindex/?folder=subfolder">subfolder/</a></td><td␣
↪align="right"> - </td><td align="right"> - </td><td align="right"> </td></tr>
<tr><td><a href="/api/access/datafile/KKKK">testfile.txt</a></td><td align="right">13-
↪January-2021 22:35</td><td align="right">19 B</td><td align="right"> </td></tr>
</table></body></html>
```

The `/dirindex/?folder=subfolder` link above will produce the following view:

Index of folder /subfolder in dataset doi:XXX/YY/ZZZZ (v. MM)

Name	Last Modified	Size	Description
50by1000.tab	11-January-2021 09:31	102.5 KB	

with the html source as follows:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html><head><title>Index of folder /subfolder</title></head>
<body><h1>Index of folder /subfolder in dataset doi:XXX/YY/ZZZZ (v. MM)</h1>
<table>
```

(continues on next page)

(continued from previous page)

```
<tr><th>Name</th><th>Last Modified</th><th>Size</th><th>Description</th></tr>
<tr><th colspan="4"><hr></th></tr>
<tr><td><a href="/api/access/datafile/subfolder/LLLL">50by1000.tab</a></td><td align=
↪"right">11-January-2021 09:31</td><td align="right">102.5 KB</td><td align="right">&
↪nbsp;</td></tr>
</table></body></html>
```

An example of a `wget` command line for crawling ("recursive downloading") of the files and folders in a dataset:

```
wget -r -e robots=off -nH --cut-dirs=3 --content-disposition https://demo.dataverse.org/
↪api/datasets/${ID}/dirindex/
# or
wget -r -e robots=off -nH --cut-dirs=3 --content-disposition https://demo.dataverse.org/
↪api/datasets/:persistentId/dirindex?persistentId=doi:${PERSISTENT_ID}
```

Note

In addition to the files and folders in the dataset, the command line above will also save the directory index of each folder, in a separate folder "dirindex".

Note

The recipe above does NOT work if the Dataverse instance serves the files via direct download (i.e., when instead of streaming the content, the download API issues redirect links to the files stored on S3). Unfortunately, this describes **every** file served by some popular Dataverse instances. With direct download, saving an entire dataset while preserving its folders structure is possible, but requires some extra scripting.

List All Metadata Blocks for a Dataset

Lists all the metadata blocks and their content, for the given dataset and version:

```
export SERVER_URL=https://demo.dataverse.org
export ID=24
export VERSION=1.0

curl "$SERVER_URL/api/datasets/$ID/versions/$VERSION/metadata"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/metadata"
```

List Single Metadata Block for a Dataset

Lists the metadata block named *METADATA_BLOCK*, for the given dataset and version:

```
export SERVER_URL=https://demo.dataverse.org
export ID=24
export VERSION=1.0
export METADATA_BLOCK=citation

curl "$SERVER_URL/api/datasets/$ID/versions/$VERSION/metadata/$METADATA_BLOCK"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/metadata/citation"
```

Compare Versions of a Dataset

Returns a list of fields that have changed between 2 Dataset versions within the Metadata and Terms of Access. Also includes the files that have been added or removed as well as files that have been modified. When compare includes an unpublished/draft version the api token must be associated with a user having view unpublished privileges An error will be returned if VERSION0 was not created before VERSION1

```
export SERVER_URL=https://demo.dataverse.org
export ID=24
export VERSION0=1.0
export VERSION1=:draft

curl "$SERVER_URL/api/datasets/$ID/versions/$VERSION0/compare/$VERSION1"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/:latest-published/compare/
↪:draft"
```

By default, deaccessioned dataset versions are not included in the search when applying the :latest or :latest-published identifiers. Additionally, when filtering by a specific version tag, you will get a "not found" error if the version is deaccessioned and you do not enable the `includeDeaccessioned` option described below.

If you want to include deaccessioned dataset versions, you must set `includeDeaccessioned` query parameter to `true`.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/:latest-published/compare/
↪:draft?includeDeaccessioned=true"
```

Get Versions of a Dataset with Summary of Changes

Returns a list of versions for a given dataset including a summary of differences between consecutive versions where available. Draft versions will only be available to users who have permission to view unpublished drafts. The api token is optional.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/BCCP9Z

curl -H "X-Dataverse-key: $API_TOKEN" -X GET "$SERVER_URL/api/datasets/:persistentId/
↪versions/compareSummary?persistentId=$PERSISTENT_IDENTIFIER"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/datasets/:persistentId/versions/compareSummary?persistentId=doi:10.
↪5072/FK2/BCCP9Z"
```

You can control pagination of the results using the following optional query parameters.

- `limit`: The maximum number of version differences to return.
- `offset`: The number of version differences to skip from the beginning of the list. Used for retrieving subsequent pages of results.

To aid in pagination the JSON response also includes the total number of rows (totalCount) available.

For example, to get the second page of results, with 2 items per page, you would use `limit=2` and `offset=2` (skipping the first two results).

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/datasets/:persistentId/versions/compareSummary?persistentId=doi:10.
↪5072/FK2/BCCP9Z&limit=2&offset=2"
```

Update Metadata For a Dataset

Updates the metadata for a dataset. If a draft of the dataset already exists, the metadata of that draft is overwritten; otherwise, a new draft is created with this metadata.

You must download a JSON representation of the dataset, edit the JSON you download, and then send the updated JSON to the Dataverse installation.

For example, after making your edits, your JSON file might look like `dataset-update-metadata.json` which you would send to the Dataverse installation like this:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/BCCP9Z

curl -H "X-Dataverse-key: $API_TOKEN" -X PUT "$SERVER_URL/api/datasets/:persistentId/
↪versions/:draft?persistentId=$PERSISTENT_IDENTIFIER" --upload-file dataset-update-
↪metadata.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/datasets/:persistentId/versions/:draft?persistentId=doi:10.5072/FK2/
↪BCCP9Z" --upload-file dataset-update-metadata.json
```

Note that in the example JSON file above, there are only two JSON objects with the `license` and `metadataBlocks` keys respectively. When you download a representation of your latest dataset version in JSON format, these objects will be nested inside another object called `data` in the API response. Note that there may be more objects in there, in addition to the `license` and `metadataBlocks` that you may need to preserve and re-import as well. Basically, you need everything in there except for the `files`. This can be achived by downloading the metadata and selecting the sections you need with a JSON tool such as `jq`, like this:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/BCCP9Z

curl -H "X-Dataverse-key: $API_TOKEN" "$SERVER_URL/api/datasets/:persistentId/versions/
↪:latest?persistentId=$PERSISTENT_IDENTIFIER" | jq '.data | del(.files)' > dataset-
↪update-metadata.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/datasets/:persistentId/versions/:latest?persistentId=doi:10.5072/FK2/BCCP9Z" |␣
↪jq '.data | {metadataBlocks: .metadataBlocks}' > dataset-update-metadata.json
```

Now you can edit the JSON produced by the command above with a text editor of your choice. For example, with `vi` in the example below.

Note that you don't need to edit the top-level fields such as `versionNumber`, `minorVersonNumber`, `versionState` or any of the time stamps - these will be automatically updated as needed by the API:

```
vi dataset-update-metadata.json
```

Now that you've made edits to the metadata in your JSON file, you can send it to a Dataverse installation as described above.

Edit Dataset Metadata

Alternatively to replacing an entire dataset version with its JSON representation you may add data to dataset fields that are blank or accept multiple values with the following:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/BCCP9Z

curl -H "X-Dataverse-key: $API_TOKEN" -X PUT "$SERVER_URL/api/datasets/:persistentId/
↪editMetadata/?persistentId=$PERSISTENT_IDENTIFIER" --upload-file dataset-add-metadata.
↪json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/datasets/:persistentId/editMetadata/?persistentId=doi:10.5072/FK2/
↪BCCP9Z" --upload-file dataset-add-metadata.json
```

You may also replace existing metadata in dataset fields with the following (adding the parameter replace=true):

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/BCCP9Z

curl -H "X-Dataverse-key: $API_TOKEN" -X PUT "$SERVER_URL/api/datasets/:persistentId/
↪editMetadata?persistentId=$PERSISTENT_IDENTIFIER&replace=true" --upload-file dataset-
↪update-metadata.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/datasets/:persistentId/editMetadata/?persistentId=doi:10.5072/FK2/
↪BCCP9Z&replace=true" --upload-file dataset-update-metadata.json
```

For these edits your JSON file need only include those dataset fields which you would like to edit. A sample JSON file may be downloaded here: `dataset-edit-metadata-sample.json`

This endpoint also allows removing fields, as long as they are not required by the dataset. To remove a field, send an empty value (`""`) for individual fields. For multiple fields, send an empty array (`[]`). A sample JSON file for removing fields may be downloaded here: `dataset-edit-metadata-delete-fields-sample.json`

If another user updates the dataset version metadata before you send the update request, metadata inconsistencies may occur. To prevent this, you can use the optional `sourceLastUpdateTime` query parameter. The intended API workflow is for the client to send along the `lastUpdateTime` obtained from the last `GET` call on the version that is being modified. Dataverse APIs will always report these time stamps in UTC, ISO 8601-formatted (`yyyy-MM-dd'T'HH:mm:ss'Z'`; for example: `2026-04-22T14:30:00Z`), regardless of the actual time zone used by the server. This is the only format this API will accept.

If this parameter is provided, the update will proceed only if the `lastUpdateTime` remains unchanged (meaning no one has updated the dataset metadata since you retrieved it). Otherwise, the request will fail with an error.

Example using `sourceLastUpdateTime`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/BCCP9Z
export SOURCE_LAST_UPDATE_TIME=2025-04-25T13:58:28Z

curl -H "X-Dataverse-key: $API_TOKEN" -X PUT "$SERVER_URL/api/datasets/:persistentId/
↪editMetadata?persistentId=$PERSISTENT_IDENTIFIER&replace=true&
↪sourceLastUpdateTime=SOURCE_LAST_UPDATE_TIME" --upload-file dataset-update-metadata.
↪json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/datasets/:persistentId/editMetadata/?persistentId=doi:10.5072/FK2/
↪BCCP9Z&replace=true&sourceLastUpdateTime=2025-04-25T13:58:28Z" --upload-file dataset-
↪update-metadata.json
```

Delete Dataset Metadata

You may delete some of the metadata of a dataset version by supplying a file with a JSON representation of dataset fields that you would like to delete with the following:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/BCCP9Z

curl -H "X-Dataverse-key: $API_TOKEN" -X PUT "$SERVER_URL/api/datasets/:persistentId/
↪deleteMetadata/?persistentId=$PERSISTENT_IDENTIFIER" --upload-file dataset-delete-
↪author-metadata.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/datasets/:persistentId/deleteMetadata/?persistentId=doi:10.5072/FK2/
↪BCCP9Z" --upload-file dataset-delete-author-metadata.json
```

For these deletes your JSON file must include an exact match of those dataset fields which you would like to delete. A sample JSON file may be downloaded here: `dataset-delete-author-metadata.json`

Update Dataset Terms of Access

Updates the terms of access for the restricted files of a dataset by applying it to the draft version, or by creating a draft if none exists.

If another user updates an already existing draft version before you send the update request, metadata inconsistencies may occur. To prevent this, you can use the optional `sourceLastUpdateTime` query parameter. The intended API workflow is for the client to send along the `lastUpdateTime` obtained from the last `GET` call on the version that is being modified. Dataverse APIs will always report these time stamps in UTC, ISO 8601-formatted (`yyyy-MM-dd'T'HH:mm:ss'Z'`; for example: `2026-04-22T14:30:00Z`), regardless of the actual time zone used by the server. This is the only format this API will accept.

If this parameter is provided, the update will proceed only if the `lastUpdateTime` remains unchanged (meaning no one has updated the dataset metadata since you retrieved it). Otherwise, the request will fail with an error.

To define custom terms of access, provide a JSON body with the following properties. All fields within `customTermsOfAccess` are optional, except if there are restricted files in your dataset then `fileAccessRequest` must be set to true or `termsOfAccess` must be provided:

```
[
  {
    "customTermsOfAccess": {
      "fileAccessRequest": true,
      "termsOfAccess": "Your terms of access for restricted files",
      "dataAccessPlace": "Your data access place",
      "originalArchive": "Your original archive",
      "availabilityStatus": "Your availability status",
      "contactForAccess": "Your contact for access",
      "sizeOfCollection": "Your size of collection",
      "studyCompletion": "Your study completion"
    }
  }
]
```

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=3
export FILE_PATH=access.json

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/datasets/$ID/access" -H
↪"Content-type:application/json" --upload-file $FILE_PATH
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/datasets/3/access" -H "Content-type:application/json" --upload-file␣
↪access.json
```

Publish a Dataset

When publishing a dataset it's good to be aware of the Dataverse Software's versioning system, which is described in the *Dataset + File Management* section of the User Guide.

If this is the first version of the dataset, its version number will be set to `1.0`. Otherwise, the new dataset version number is determined by the most recent version number and the `type` parameter. Passing `type=minor` increases the minor version number (2.3 is updated to 2.4). Passing `type=major` increases the major version number (2.3 is updated to 3.0). (Superusers can pass `type=updatecurrent` to update metadata without changing the version number.)

This call also supports an optional boolean query parameter: `assureIsIndexed`. If true, the call will fail with a 409 ("CONFLICT") response if the dataset is awaiting re-indexing. If indexing occurs during publishing it could cause the publish request to fail, after a 202 response has been received. Using this parameter allows the caller to wait for indexing to occur and avoid this possibility. It is most useful in situations where edits are made immediately before publication.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB
export MAJOR_OR_MINOR=major

curl -H "X-Dataverse-key: $API_TOKEN" -X POST "$SERVER_URL/api/datasets/:persistentId/
↪actions/:publish?persistentId=$PERSISTENT_ID&type=$MAJOR_OR_MINOR"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/datasets/:persistentId/actions/:publish?persistentId=doi:10.5072/FK2/
↪J8SJZB&type=major"
```

The quotes around the URL are required because there is more than one query parameter separated by an ampersand (&), which has special meaning to Unix shells such as Bash. Putting the & in quotes ensures that "type" is interpreted as one of the query parameters.

You should expect JSON output and a 200 ("OK") response in most cases. If you receive a 202 ("ACCEPTED") response, this is normal for installations that have workflows configured. Workflows are described in the *Workflows* section of the Developer Guide. A 409 ("CONFLICT") response is also possible if you set `assureIsIndexed=true`. (In this case, one could then repeat the call until a 200/202 response is sent.)

Note

POST should be used to publish a dataset. GET is supported for backward compatibility but is deprecated and may be removed: https://github.com/IQSS/dataverse/issues/2431

Delete Dataset Draft

Deletes the draft version of dataset `$ID`. Only the draft version can be deleted:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE "$SERVER_URL/api/datasets/$ID/versions/
↪:draft"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/datasets/24/versions/:draft"
```

Deaccession Dataset

Given a version of a dataset, updates its status to deaccessioned.

The JSON body required to deaccession a dataset (`deaccession.json`) looks like this:

```
{
  "deaccessionReason": "Description of the deaccession reason.",
  "deaccessionForwardURL": "https://demo.dataverse.org"
}
```

Note that the field `deaccessionForwardURL` is optional.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24
export VERSIONID=1.0
export FILE_PATH=deaccession.json

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/datasets/$ID/versions/
↪$VERSIONID/deaccession" -H "Content-type:application/json" --upload-file $FILE_PATH
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/datasets/24/versions/1.0/deaccession" -H "Content-type:application/
↪json" --upload-file deaccession.json
```

Note

You cannot deaccession a dataset more than once. If you call this endpoint twice for the same dataset version, you will get a not found error on the second call, since the dataset you are looking for will no longer be published since it is already deaccessioned.

Set Citation Date Field Type for a Dataset

Sets the dataset citation date field type for a given dataset. `:publicationDate` is the default. Note that the dataset citation date field type must be a date field. This change applies to all versions of the dataset that have an entry for the new date field. It also applies to all file citations in the dataset.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB
export DATASET_FIELD_TYPE_NAME=dateOfDeposit

curl -H "X-Dataverse-key: $API_TOKEN" -X PUT "$SERVER_URL/api/datasets/:persistentId/
↪citationdate?persistentId=$PERSISTENT_IDENTIFIER" --data "$DATASET_FIELD_TYPE_NAME"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/datasets/:persistentId/citationdate?persistentId=doi:10.5072/FK2/
↪J8SJZB" --data "dateOfDeposit"
```

Revert Citation Date Field Type to Default for Dataset

Restores the default citation date field type, `:publicationDate`, for a given dataset.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB

curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE "$SERVER_URL/api/datasets/:persistentId/
↪citationdate?persistentId=$PERSISTENT_IDENTIFIER"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/datasets/:persistentId/citationdate?persistentId=doi:10.5072/FK2/
↪J8SJZB"
```

List Role Assignments in a Dataset

Lists all role assignments on a given dataset:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=2347

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/datasets/$ID/assignments"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  "https://demo.dataverse.
↪org/api/datasets/2347/assignments"
```

Assign a New Role on a Dataset

Assigns a new role, based on the POSTed JSON:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=2347

curl -H "X-Dataverse-key:$API_TOKEN" -X POST -H "Content-Type: application/json" "
↪$SERVER_URL/api/datasets/$ID/assignments" --upload-file role.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST -H "Content-Type:␣
↪application/json" "https://demo.dataverse.org/api/datasets/2347/assignments" --upload-
↪file role.json
```

POSTed JSON example (the content of `role.json` file):

```
{
  "assignee": "@uma",
  "role": "curator"
}
```

Delete Role Assignment from a Dataset

Delete the assignment whose id is `$id`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=2347
export ASSIGNMENT_ID=6

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/datasets/$ID/assignments/
↪$ASSIGNMENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/datasets/2347/assignments/6"
```

Create a Preview URL for a Dataset

Create a Preview URL (must be able to manage dataset permissions):

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key: $API_TOKEN" -X POST "$SERVER_URL/api/datasets/$ID/previewUrl"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/datasets/24/previewUrl"
```

If Anonymized Access has been enabled on a Dataverse installation (see the *:AnonymizedFieldTypeNames* setting), an optional 'anonymizedAccess' query parameter is allowed. Setting anonymizedAccess=true in your call will create a PreviewURL that only allows an anonymized view of the Dataset (see *Preview URL to Review Unpublished Dataset*).

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/datasets/24/previewUrl?anonymizedAccess=true"
```

Note: Previous endpoints with privateUrl instead of previewUrl are deprecated, but supported.

Get the Preview URL for a Dataset

Get a Preview URL from a dataset (if available):

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key: $API_TOKEN" "$SERVER_URL/api/datasets/$ID/previewUrl"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/datasets/24/previewUrl"
```

Delete the Preview URL from a Dataset

Delete a Preview URL from a dataset (if it exists):

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE "$SERVER_URL/api/datasets/$ID/previewUrl"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/datasets/24/previewUrl"
```

Add a File to a Dataset

When adding a file to a dataset, you can optionally specify the following:

- A description of the file.
- The "File Path" of the file, indicating which folder the file should be uploaded to within the dataset.
- Whether or not the file is restricted.
- Whether or not the file skips *tabular ingest*. If the `tabIngest` parameter is not specified, it defaults to `true`.

Note that when a Dataverse installation is configured to use S3 storage with direct upload enabled, there is API support to send a file directly to S3. This is more complex and is described in the *Direct DataFile Upload/Replace API* guide. Also, see *Setting the files count limit on a Dataset*, for limitations to the number of files allowed per Dataset.

In the curl example below, all of the above are specified but they are optional.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export FILENAME='data.tsv'
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB

curl -H "X-Dataverse-key:$API_TOKEN" -X POST -F "file=@$FILENAME" -F 'jsonData={
↪"description":"My description.","directoryLabel":"data/subdir1","categories":["Data"],
↪"restrict":"false", "tabIngest":"false"}' "$SERVER_URL/api/datasets/:persistentId/add?
↪persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST -F file=@data.tsv␣
↪-F 'jsonData={"description":"My description.","directoryLabel":"data/subdir1",
↪"categories":["Data"], "restrict":"false", "tabIngest":"false"}' "https://demo.
↪dataverse.org/api/datasets/:persistentId/add?persistentId=doi:10.5072/FK2/J8SJZB"
```

You should expect a 201 ("CREATED") response and JSON indicating the database id that has been assigned to your newly uploaded file.

Please note that it's possible to "trick" a Dataverse installation into giving a file a content type (MIME type) of your choosing. For example, you can make a text file be treated like a video file with `-F 'file=@README.txt; type=video/mpeg4'`, for example. If the Dataverse installation does not properly detect a file type, specifying the content type via API like this a potential workaround.

The curl syntax above to upload a file is tricky and a Python version is provided below. (Please note that it depends on libraries such as "requests" that you may need to install but this task is out of scope for this guide.) Here are some parameters you can set in the script:

- `dataverse_server` - e.g. https://demo.dataverse.org
- `api_key` - See the top of this document for a description
- `persistentId` - Example: `doi:10.5072/FK2/6XACVA`
- `dataset_id` - Database id of the dataset

In practice, you only need one the `dataset_id` or the `persistentId`. The example below shows both uses.

```
from datetime import datetime
import json
import requests  # http://docs.python-requests.org/en/master/

# ------------------------------------------------
# Update the 4 params below to run this code
# ------------------------------------------------
dataverse_server = 'https://your dataverse installation' # no trailing slash
api_key = 'api key'
dataset_id = 1  # database id of the dataset
persistentId = 'doi:10.5072/FK2/6XACVA' # doi or hdl of the dataset

# ------------------------------------------------
# Prepare "file"
# ------------------------------------------------
file_content = 'content: %s' % datetime.now()
files = {'file': ('sample_file.txt', file_content)}

# ------------------------------------------------
# Using a "jsonData" parameter, add optional description + file tags
# ------------------------------------------------
params = dict(description='Blue skies!',
              categories=['Lily', 'Rosemary', 'Jack of Hearts'])

params_as_json_string = json.dumps(params)

payload = dict(jsonData=params_as_json_string)

# ------------------------------------------------
# Add file using the Dataset's id
# ------------------------------------------------
url_dataset_id = '%s/api/datasets/%s/add?key=%s' % (dataverse_server, dataset_id, api_
↪key)

# -------------------
```

(continues on next page)

(continued from previous page)

```
# Make the request
# -------------------
print '-' * 40
print 'making request: %s' % url_dataset_id
r = requests.post(url_dataset_id, data=payload, files=files)

# -------------------
# Print the response
# -------------------
print '-' * 40
print r.json()
print r.status_code

# ------------------------------------------------
# Add file using the Dataset's persistentId (e.g. doi, hdl, etc)
# ------------------------------------------------
url_persistent_id = '%s/api/datasets/:persistentId/add?persistentId=%s&key=%s' %␣
↪(dataverse_server, persistentId, api_key)

# -------------------
# Update the file content to avoid a duplicate file error
# -------------------
file_content = 'content2: %s' % datetime.now()
files = {'file': ('sample_file2.txt', file_content)}


# -------------------
# Make the request
# -------------------
print '-' * 40
print 'making request: %s' % url_persistent_id
r = requests.post(url_persistent_id, data=payload, files=files)

# -------------------
# Print the response
# -------------------
print '-' * 40
print r.json()
print r.status_code
```

Add a Remote File to a Dataset

If your Dataverse installation has been configured to support *Trusted Remote Storage* you can add files from remote URLs to datasets. These remote files appear in your Dataverse installation as if they were ordinary files but are stored remotely.

The location of the remote file is specified in the `storageIdentifier` field in JSON you supply. The base URL of the file is contained in the "store" (e.g. "trsa" in the example below) and is followed by the path to the file (e.g. "themes/custom…").

In the JSON example below, all fields are required except for `description`. Other optional fields are shown under *Add a File to a Dataset*.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB
export JSON_DATA='{"description":"A remote image.","storageIdentifier":"trsa://themes/
↪custom/qdr/images/CoreTrustSeal-logo-transparent.png","checksumType":"MD5","md5Hash":
↪"509ef88afa907eaf2c17c1c8d8fde77e","label":"testlogo.png","fileName":"testlogo.png",
↪"mimeType":"image/png"}'

curl -H "X-Dataverse-key: $API_TOKEN" -X POST "$SERVER_URL/api/datasets/:persistentId/
↪add?persistentId=$PERSISTENT_ID" -F "jsonData=$JSON_DATA"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/datasets/:persistentId/add?persistentId=doi:10.5072/FK2/J8SJZB" -F
↪'jsonData={"description":"A remote image.","storageIdentifier":"trsa://themes/custom/
↪qdr/images/CoreTrustSeal-logo-transparent.png","checksumType":"MD5","md5Hash":
↪"509ef88afa907eaf2c17c1c8d8fde77e","label":"testlogo.png","fileName":"testlogo.png",
↪"mimeType":"image/png"}'
```

Update File Metadata

Updates metadata for one or more files in a dataset. This API call allows you to modify file-level metadata without the need to replace the actual file content.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/datasets/:persistentId/
↪files/metadata?persistentId=$PERSISTENT_ID" --upload-file file-metadata-update.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/datasets/:persistentId/files/metadata?:persistentId=doi:10.5072/FK2/
↪J8SJZB" --upload-file file-metadata-update.json
```

The `file-metadata-update.json` file should contain a JSON array of objects, each representing a file to be updated. Here's an example structure:

```
[
  {
    "dataFileId": 42,
    "label": "Updated File Name",
    "directoryLabel": "data/",
    "description": "Updated file description",
    "restricted": false,
    "categories": ["Documentation", "Data"],
    "provFreeForm": "Updated provenance information"
  },
  {
```

(continues on next page)

(continued from previous page)

```
    "dataFileId": 43,
    "label": "Another Updated File",
    "description": "Another updated description",
    "restricted": true
  }
]
```

Each object in the array must include the `dataFileId` field to identify the file. Other fields are optional and will only be updated if included.

The API will return a JSON object with information about the update operation, including any errors that occurred during the process.

Note: This API call requires appropriate permissions to edit the dataset and its files.

Cleanup Storage of a Dataset

This is an experimental feature and should be tested on your system before using it in production. Also, make sure that your backups are up-to-date before using this on production servers. It is advised to first call this method with the `dryrun` parameter set to `true` before actually deleting the files. This will allow you to manually inspect the files that would be deleted if that parameter is set to `false` or is omitted (a list of the files that would be deleted is provided in the response).

If your Dataverse installation has been configured to support direct uploads, or in some other situations, you could end up with some files in the storage of a dataset that are not linked to that dataset directly. Most commonly, this could happen when an upload fails in the middle of a transfer, i.e. if a user does a UI direct upload and leaves the page without hitting cancel or save, Dataverse doesn't know and doesn't clean up the files. Similarly in the direct upload API, if the final /addFiles call isn't done, the files are abandoned.

All the files stored in the Dataset storage location that are not in the file list of that Dataset (and follow the naming pattern of the dataset files) can be removed, as shown in the example below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB
export DRYRUN=true

curl -H "X-Dataverse-key: $API_TOKEN" -X GET "$SERVER_URL/api/datasets/:persistentId/
↪cleanStorage?persistentId=$PERSISTENT_ID&dryrun=$DRYRUN"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/datasets/:persistentId/cleanStorage?persistentId=doi:10.5072/FK2/
↪J8SJZB&dryrun=true"
```

Adding Files To a Dataset via Other Tools

In some circumstances, it may be useful to move or copy files into Dataverse's storage manually or via external tools and then add then to a dataset (i.e. without involving Dataverse in the file transfer itself). Two API calls are available for this use case to add files to a dataset or to replace files that were already in the dataset. These calls were developed as part of Dataverse's direct upload mechanism and are detailed in *Direct DataFile Upload/Replace API*.

Imposing a limit to the number of files allowed to be uploaded to a Dataset

Having thousands of files in a Dataset can cause issues. Most users would be better off with the data repackaged in fewer large bundles. To help curtail these issues, a limit can be set to prevent the number of file uploads from getting out of hand.

The limit can be set via JVM setting *dataverse.files.default-dataset-file-count-limit* to be installation wide, or, set on each Collection/Dataset.

For Installation wide limit, the limit can be set via JVM. ./asadmin $ASADMIN_OPTS create-jvm-options "-Ddataverse.files.default-dataset-file-count-limit=<limit>"

For Collections, the attribute can be controlled by calling the Create or Update Dataverse API and adding `datasetFileCountLimit=500` to the Json body (Must be a superuser to change this value).

For Datasets, the attribute can be set using the *Update Dataset Files Limit* API and passing the qp *fileCountLimit=500* (Must be a superuser to change this value).

Setting a value of -1 will clear the limit for that level. If no limit is found on the Dataset, the hierarchy of parent nodes will be checked until finally the JVM setting is checked.

With this setting set a 400 error response stating that the limit has been reached, including the effective limit, will be returned.

Please note that a superuser will be exempt from this rule.

The check will use the value defined in the Dataset first, and if not set (value <1) the Dataverse/Collection will be checked, and finally the JVM setting.

Setting the files count limit on a Dataset

In order to update the number of files allowed for a Dataset, without causing a Draft version of the Dataset being created, the following API can be used

> **Note**
>
> To clear the limit simply set the limit to -1 or call the DELETE API (Must be a superuser to change this value).

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24
export LIMIT=500

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/datasets/$ID/files/
↪uploadlimit/$LIMIT"
```

To delete the existing limit:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/datasets/$ID/files/
↪uploadlimit"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/datasets/24/files/uploadlimit/500"
```

To delete the existing limit:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/datasets/24/files/uploadlimit"
```

Report the data (file) size of a Dataset

Shows the combined size in bytes of all the files uploaded into the dataset `id`.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/datasets/$ID/storagesize"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/datasets/24/storagesize"
```

The size of published and unpublished files will be summed in the dataset specified. By default, only the archival files are counted - i.e., the files uploaded by users (plus the tab-delimited versions generated for tabular data files on ingest). If the optional argument `includeCached=true` is specified, the API will also add the sizes of all the extra files generated and cached by the Dataverse installation - the resized thumbnail versions for image files, the metadata exports for published datasets, etc. Because this deals with unpublished files the token supplied must have permission to view unpublished drafts.

Get the size of Downloading all the files of a Dataset Version

Shows the combined size in bytes of all the files available for download from version `versionId` of dataset `id`.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24
export VERSIONID=1.0

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/datasets/$ID/versions/$VERSIONID/
↪downloadsize"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/datasets/24/versions/1.0/downloadsize"
```

The size of all files available for download will be returned. If :draft is passed as versionId the token supplied must have permission to view unpublished drafts. A token is not required for published datasets. Also restricted files will be included in this total regardless of whether the user has access to download the restricted file(s).

There is an optional query parameter `mode` which applies a filter criteria to the operation. This parameter supports the following values:

- `All` (Default): Includes both archival and original sizes for tabular files
- `Archival`: Includes only the archival size for tabular files
- `Original`: Includes only the original size for tabular files

Usage example:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/datasets/24/versions/1.0/downloadsize?mode=Archival"
```

Category name filtering is also optionally supported. To return the size of all files available for download matching the requested category name.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/downloadsize?
↪categoryName=Data"
```

Tabular tag name filtering is also optionally supported. To return the size of all files available for download for which the requested tabular tag has been added.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/downloadsize?
↪tabularTagName=Survey"
```

Content type filtering is also optionally supported. To return the size of all files available for download matching the requested content type.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/downloadsize?
↪contentType=image/png"
```

Filtering by search text is also optionally supported. The search will be applied to the labels and descriptions of the dataset files, to return the size of all files available for download that contain the text searched in one of such fields.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/downloadsize?
↪searchText=word"
```

File access filtering is also optionally supported. In particular, by the following possible values:

- `Public`
- `Restricted`
- `EmbargoedThenRestricted`
- `EmbargoedThenPublic`
- `RetentionPeriodExpired`

If no filter is specified, the files will match all of the above categories.

Please note that filtering query parameters are case sensitive and must be correctly typed for the endpoint to recognize them.

By default, deaccessioned dataset versions are not included in the search when applying the :latest or :latest-published identifiers. Additionally, when filtering by a specific version tag, you will get a "not found" error if the version is deaccessioned and you do not enable the `includeDeaccessioned` option described below.

If you want to include deaccessioned dataset versions, you must set `includeDeaccessioned` query parameter to `true`.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/24/versions/1.0/downloadsize?
↪includeDeaccessioned=true"
```

Note

Keep in mind that you can combine all of the above query parameters depending on the results you are looking for.

Get the Download count of a Dataset

Shows the total number of downloads requested for a dataset. If MDC is enabled the count will be limited to the time before MDC start if the optional *includeMDC* parameter is not included or set to False. Setting *includeMDC* to True will ignore the *:MDCStartDate* setting and return a total count.

```
API_TOKEN='xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
export DATASET_ID=1
export includeMDC=True

curl -s -H "X-Dataverse-key:$API_TOKEN" -X GET http://localhost:8080/api/datasets/
↪$DATASET_ID/download/count
curl -s -H "X-Dataverse-key:$API_TOKEN" -X GET http://localhost:8080/api/datasets/
↪$DATASET_ID/download/count?includeMDC=true
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/datasets/1/download/count?includeMDC=False"
```

Submit a Dataset for Review

When dataset authors do not have permission to publish directly, they can click the "Submit for Review" button in the web interface (see *Dataset + File Management*), or perform the equivalent operation via API:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB

curl -H "X-Dataverse-key: $API_TOKEN" -X POST "$SERVER_URL/api/datasets/:persistentId/
↪submitForReview?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/datasets/:persistentId/submitForReview?persistentId=doi:10.5072/FK2/
↪J8SJZB"
```

The people who need to review the dataset (often curators or journal editors) can check their notifications periodically via API to see if any new datasets have been submitted for review and need their attention. See the *Notifications* section for details. Alternatively, these curators can simply check their email or notifications to know when datasets have been submitted (or resubmitted) for review.

Return a Dataset to Author

After the curators or journal editors have reviewed a dataset that has been submitted for review (see "Submit for Review", above) they can either choose to publish the dataset (see the `:publish` "action" above) or return the dataset to its authors. In the web interface there is a "Return to Author" button (see *Dataset + File Management*). The same operation can be done via this API call.

Here's how curators can send a "reason for return" to the dataset authors. First, the curator creates a JSON file that contains the reason for return:

```
{
  "reasonForReturn": "You forgot to upload any files."
}
```

In the example below, the curator has saved the JSON file as `reason-for-return.json` in their current working directory. Then, the curator sends this JSON file to the `returnToAuthor` API endpoint like this:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB

curl -H "X-Dataverse-key: $API_TOKEN" -X POST "$SERVER_URL/api/datasets/:persistentId/
↪returnToAuthor?persistentId=$PERSISTENT_ID" -H "Content-type: application/json" -d␣
↪@reason-for-return.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/datasets/:persistentId/returnToAuthor?persistentId=doi:10.5072/FK2/
↪J8SJZB" -H "Content-type: application/json" -d @reason-for-return.json
```

The review process can sometimes resemble a tennis match, with the authors submitting and resubmitting the dataset over and over until the curators are satisfied. Each time the curators send a "reason for return" via API, that reason is sent by email and is persisted into the database, stored at the dataset version level. Note the reason is required, unless the *dataverse.feature.disable-return-to-author-reason* feature flag has been set. Reason is a free text field and could be as simple as "The author would like to modify his dataset", "Files are missing", "Nothing to report" or "A curation report with comments and suggestions/instructions will follow in another email" that suits your situation.

The *Send Feedback To Contact(s) Admin API* Admin only API call may be useful as a way to move the conversation to email. However, note that these emails go to contacts (versus authors) and there is no database record of the email contents. (*dataverse.mail.cc-support-on-contact-email* will send a copy of these emails to the support email address which would provide a record.) The *Send Feedback To Contact(s)* API call may be useful as a way to move the conversation to email. However, note that these emails go to contacts (versus authors) and there is no database record of the email contents. (*dataverse.mail.cc-support-on-contact-email* will send a copy of these emails to the support email address which would provide a record.)

Link a Dataset

Creates a link between a dataset and a Dataverse collection (see *Dataset Linking* section of Dataverse Collection Management in the User Guide for more information):

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export DATASET_ID=24
export DATAVERSE_ID=test

curl -H "X-Dataverse-key: $API_TOKEN" -X PUT "$SERVER_URL/api/datasets/$DATASET_ID/link/
↪$DATAVERSE_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/datasets/24/link/test"
```

Dataset Locks

Manage Locks on a Specific Dataset

To check if a dataset is locked:

```
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl "$SERVER_URL/api/datasets/$ID/locks"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/24/locks"
```

Optionally, you can check if there's a lock of a specific type on the dataset:

```
export SERVER_URL=https://demo.dataverse.org
export ID=24
export LOCK_TYPE=Ingest

curl "$SERVER_URL/api/datasets/$ID/locks?type=$LOCK_TYPE"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/24/locks?type=Ingest"
```

Currently implemented lock types are `Ingest`, `Workflow`, `InReview`, `DcmUpload` (deprecated), `finalizePublication`, `EditInProgress` and `FileValidationFailed`.

The API will output the list of locks, for example:

```
{"status":"OK","data":
  [
    {
```

(continues on next page)

(continued from previous page)

```
      "lockType":"Ingest",
      "date":"Fri Aug 17 15:05:51 EDT 2018",
      "user":"dataverseAdmin",
      "dataset":"doi:12.34567/FK2/ABCDEF"
    },
    {
      "lockType":"Workflow",
      "date":"Fri Aug 17 15:02:00 EDT 2018",
      "user":"dataverseAdmin",
      "dataset":"doi:12.34567/FK2/ABCDEF"
    }
  ]
}
```

If the dataset is not locked (or if there is no lock of the requested type), the API will return an empty list.

The following API end point will lock a Dataset with a lock of specified type. Note that this requires "superuser" credentials:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24
export LOCK_TYPE=Ingest

curl -H "X-Dataverse-key: $API_TOKEN" -X POST "$SERVER_URL/api/datasets/$ID/lock/$LOCK_
↪TYPE"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/datasets/24/lock/Ingest"
```

Use the following API to unlock the dataset, by deleting all the locks currently on the dataset. Note that this requires "superuser" credentials:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE "$SERVER_URL/api/datasets/$ID/locks"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/datasets/24/locks"
```

Or, to delete a lock of the type specified only. Note that this requires "superuser" credentials:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24
export LOCK_TYPE=finalizePublication
```

(continues on next page)

(continued from previous page)

```
curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE "$SERVER_URL/api/datasets/$ID/locks?type=
↪$LOCK_TYPE"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/datasets/24/locks?type=finalizePublication"
```

If the dataset is not locked (or if there is no lock of the specified type), the API will exit with a warning message.

(Note that the API calls above all support both the database id and persistent identifier notation for referencing the dataset)

List Locks Across All Datasets

Note that this API requires "superuser" credentials. You must supply the `X-Dataverse-key` header with the api token of an admin user (as in the example below).

The output of this API is formatted identically to the API that lists the locks for a specific dataset, as in one of the examples above.

Use the following API to list ALL the locks on all the datasets in your installation:

```
/api/datasets/locks
```

The listing can be filtered by specific lock type **and/or** user, using the following *optional* query parameters:

- `userIdentifier` - To list the locks owned by a specific user
- `type` - To list the locks of the type specified. If the supplied value does not match a known lock type, the API will return an error and a list of valid lock types. As of writing this, the implemented lock types are `Ingest`, `Workflow`, `InReview`, `DcmUpload` (deprecated), `finalizePublication`, `EditInProgress` and `FileValidationFailed`.

For example:

```
curl -H "X-Dataverse-key: xxx" "http://localhost:8080/api/datasets/locks?type=Ingest&
↪userIdentifier=davis4ever"
```

Dataset Metrics

Please note that these dataset level metrics are only available if support for Make Data Count has been enabled in your Dataverse installation. See the *Dataset Metrics* in the *Dataset + File Management* section of the User Guide and the *Make Data Count* section of the Admin Guide for details.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

```
export SERVER_URL=https://demo.dataverse.org
```

To confirm that the environment variable was set properly, you can use `echo` like this:

```
echo $SERVER_URL
```

Please note that for each of these endpoints except the "citations" endpoint, you can optionally pass the query parameter "country" with a two letter code (e.g. "country=us") and you can specify a particular month by adding it in yyyy-mm format after the requested metric (e.g. "viewsTotal/2019-02").

Retrieving Total Views for a Dataset

Please note that "viewsTotal" is a combination of "viewsTotalRegular" and "viewsTotalMachine" which can be requested separately.

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB

curl "$SERVER_URL/api/datasets/:persistentId/makeDataCount/viewsTotal?persistentId=
↪$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/:persistentId/makeDataCount/viewsTotal?
↪persistentId=10.5072/FK2/J8SJZB"
```

Retrieving Unique Views for a Dataset

Please note that "viewsUnique" is a combination of "viewsUniqueRegular" and "viewsUniqueMachine" which can be requested separately.

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB

curl "$SERVER_URL/api/datasets/:persistentId/makeDataCount/viewsUnique?persistentId=
↪$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/:persistentId/makeDataCount/viewsUnique?
↪persistentId=10.5072/FK2/J8SJZB"
```

Retrieving Total Downloads for a Dataset

Please note that "downloadsTotal" is a combination of "downloadsTotalRegular" and "downloadsTotalMachine" which can be requested separately.

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB

curl "$SERVER_URL/api/datasets/:persistentId/makeDataCount/downloadsTotal?persistentId=
↪$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/:persistentId/makeDataCount/downloadsTotal?
↪persistentId=10.5072/FK2/J8SJZB"
```

Retrieving Unique Downloads for a Dataset

Please note that "downloadsUnique" is a combination of "downloadsUniqueRegular" and "downloadsUniqueMachine" which can be requested separately.

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB

curl "$SERVER_URL/api/datasets/:persistentId/makeDataCount/downloadsUnique?persistentId=
↪$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/:persistentId/makeDataCount/
↪downloadsUnique?persistentId=10.5072/FK2/J8SJZB"
```

Retrieving Citations for a Dataset

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB

curl "$SERVER_URL/api/datasets/:persistentId/makeDataCount/citations?persistentId=
↪$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/:persistentId/makeDataCount/citations?
↪persistentId=10.5072/FK2/J8SJZB"
```

Delete Unpublished Dataset

Delete the dataset whose id is passed:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE "$SERVER_URL/api/datasets/$ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/datasets/24"
```

Delete Published Dataset

Normally published datasets should not be deleted, but there exists a "destroy" API endpoint for superusers which will act on a dataset given a persistent ID or dataset database ID:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE "$SERVER_URL/api/datasets/:persistentId/
↪destroy/?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/datasets/:persistentId/destroy/?persistentId=doi:10.5072/FK2/AAA000"
```

Delete with dataset identifier:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE "$SERVER_URL/api/datasets/$ID/destroy"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/datasets/24/destroy"
```

Calling the destroy endpoint is permanent and irreversible. It will remove the dataset and its datafiles, then re-index the parent Dataverse collection in Solr. This endpoint requires the API token of a superuser.

Configure a Dataset to Use a Specific File Store

`/api/datasets/$dataset-id/storageDriver` can be used to check, configure or reset the designated file store (storage driver) for a dataset. Please see the *Managing Datasets and Dataverse Collections* section of the guide for more information on this API.

View the Timestamps on a Dataset

`/api/datasets/$dataset-id/timestamps` can be used to view timestamps associated with various events in the dataset's lifecycle. For published datasets, this API call provides the `createTime`, `publicationTime`, `lastMetadataExportTime` and `lastMajorVersionReleaseTime`, as well as two booleans - `hasStaleIndex` and `hasStalePermissionIndex` - which, if false, indicate the Dataverse displays for the dataset are up-to-date. The response is `application/json` with the timestamps included in the returned `data` object.

When called by a user who can view the draft version of the dataset, additional timestamps are reported: `lastUpdateTime`, `lastIndexTime`, `lastPermissionUpdateTime`, and `globalIdCreateTime`.

One use case where this API call could be useful is in allowing an external application to poll and wait for changes being made by the Dataverse software or other external tool to complete prior to continuing its own processing.

Set an Embargo on Files in a Dataset

/api/datasets/$dataset-id/files/actions/:set-embargo can be used to set an embargo on one or more files in a dataset. Embargoes can be set on files that are only in a draft dataset version (and are not in any previously published version) by anyone who can edit the dataset. The same API call can be used by a superuser to add an embargo to files that have already been released as part of a previously published dataset version.

The API call requires a Json body that includes the embargo's end date (dateAvailable - YYYY-MM-DD format), a short reason (must not consist of whitespace only, optional unless Dataverse is configured to make it required), and a list of the fileIds that the embargo should be set on. The dateAvailable must be after the current date and the duration (dateAvailable - today's date) must be less than the value specified by the *:MaxEmbargoDurationInMonths* setting. All files listed must be in the specified dataset. For example:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV
export JSON='{"dateAvailable":"2021-01-20", "reason":"Standard project embargo", "fileIds
↪":[300,301,302]}'

curl -H "X-Dataverse-key: $API_TOKEN" -H "Content-Type:application/json" "$SERVER_URL/
↪api/datasets/:persistentId/files/actions/:set-embargo?persistentId=$PERSISTENT_
↪IDENTIFIER" -d "$JSON"
```

Remove an Embargo on Files in a Dataset

/api/datasets/$dataset-id/files/actions/:unset-embargo can be used to remove an embargo on one or more files in a dataset. Embargoes can be removed from files that are only in a draft dataset version (and are not in any previously published version) by anyone who can edit the dataset. The same API call can be used by a superuser to remove embargos from files that have already been released as part of a previously published dataset version.

The API call requires a Json body that includes the list of the fileIds that the embargo should be removed from. All files listed must be in the specified dataset. For example:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV
export JSON='{"fileIds":[300,301]}'

curl -H "X-Dataverse-key: $API_TOKEN" -H "Content-Type:application/json" "$SERVER_URL/
↪api/datasets/:persistentId/files/actions/:unset-embargo?persistentId=$PERSISTENT_
↪IDENTIFIER" -d "$JSON"
```

Set a Retention Period on Files in a Dataset

/api/datasets/$dataset-id/files/actions/:set-retention can be used to set a retention period on one or more files in a dataset. Retention periods can be set on files that are only in a draft dataset version (and are not in any previously published version) by anyone who can edit the dataset. The same API call can be used by a superuser to add a retention period to files that have already been released as part of a previously published dataset version.

The API call requires a Json body that includes the retention period's end date (dateUnavailable), a short reason (optional), and a list of the fileIds that the retention period should be set on. The dateUnavailable must be after the current date and the duration (dateUnavailable - today's date) must be larger than the value specified by the *:MinRetentionDurationInMonths* setting. All files listed must be in the specified dataset. For example:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV
export JSON='{"dateUnavailable":"2051-12-31", "reason":"Standard project retention period
↪", "fileIds":[300,301,302]}'

curl -H "X-Dataverse-key: $API_TOKEN" -H "Content-Type:application/json" "$SERVER_URL/
↪api/datasets/:persistentId/files/actions/:set-retention?persistentId=$PERSISTENT_
↪IDENTIFIER" -d "$JSON"
```

Remove a Retention Period on Files in a Dataset

`/api/datasets/$dataset-id/files/actions/:unset-retention` can be used to remove a retention period on one or more files in a dataset. Retention periods can be removed from files that are only in a draft dataset version (and are not in any previously published version) by anyone who can edit the dataset. The same API call can be used by a superuser to remove retention periods from files that have already been released as part of a previously published dataset version.

The API call requires a Json body that includes the list of the fileIds that the retention period should be removed from. All files listed must be in the specified dataset. For example:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV
export JSON='{"fileIds":[300,301]}'

curl -H "X-Dataverse-key: $API_TOKEN" -H "Content-Type:application/json" "$SERVER_URL/
↪api/datasets/:persistentId/files/actions/:unset-retention?persistentId=$PERSISTENT_
↪IDENTIFIER" -d "$JSON"
```

Get the Archival Status of a Dataset By Version

Archival *BagIt Export* is an optional feature that may be configured for a Dataverse installation. When that is enabled, this API call be used to retrieve the status. Note that this requires "superuser" credentials.

`GET /api/datasets/$dataset-id/$version/archivalStatus` returns the archival status of the specified dataset version.

The response is a JSON object that will contain a "status" which may be "success", "pending", or "failure" and a "message" which is archive system specific. For "success" the message should provide an identifier or link to the archival copy. For example:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV
export VERSION=1.0

curl -H "X-Dataverse-key: $API_TOKEN" -H "Accept:application/json" "$SERVER_URL/api/
↪datasets/:persistentId/$VERSION/archivalStatus?persistentId=$PERSISTENT_IDENTIFIER"
```

Set the Archival Status of a Dataset By Version

Archiving is an optional feature that may be configured for a Dataverse installation. When that is enabled, this API call be used to set the status. Note that this is intended to be used by the archival system and requires "superuser" credentials.

`PUT /api/datasets/$dataset-id/$version/archivalStatus` sets the archival status of the specified dataset version.

The body is a JSON object that must contain a "status" which may be "success", "pending", or "failure" and a "message" which is archive system specific. For "success" the message should provide an identifier or link to the archival copy. For example:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV
export VERSION=1.0
export JSON='{"status":"failure","message":"Something went wrong"}'

curl -H "X-Dataverse-key: $API_TOKEN" -H "Content-Type:application/json" -X PUT "$SERVER_
↪URL/api/datasets/:persistentId/$VERSION/archivalStatus?persistentId=$PERSISTENT_
↪IDENTIFIER" -d "$JSON"
```

Note that if the configured archiver only supports archiving a single version, the call may return 409 CONFLICT if/when another version already has a non-null status.

Delete the Archival Status of a Dataset By Version

Archiving is an optional feature that may be configured for a Dataverse installation. When that is enabled, this API call be used to delete the status. Note that this is intended to be used by the archival system and requires "superuser" credentials.

`DELETE /api/datasets/$dataset-id/$version/archivalStatus` deletes the archival status of the specified dataset version.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV
export VERSION=1.0

curl -H "X-Dataverse-key: $API_TOKEN" -X DELETE "$SERVER_URL/api/datasets/:persistentId/
↪$VERSION/archivalStatus?persistentId=$PERSISTENT_IDENTIFIER"
```

Get Dataset External Tool URL

This API call generates a URL for accessing an external tool (see *External Tools*) that operates at the dataset level. The URL includes necessary authentication tokens and parameters based on the user's permissions and the tool's configuration.

Authentication is required for draft or deaccessioned datasets and the user must have ViewUnpublishedDataset permission.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
```

(continues on next page)

(continued from previous page)

```
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV
export TOOL_ID=42

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/datasets/:persistentId/
↪externalTool/$TOOL_ID/toolUrl?persistentId=$PERSISTENT_IDENTIFIER" \
-H "Content-Type: application/json" \
-d '{"preview": false, "locale": "en"}'
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/datasets/:persistentId/externalTool/42/toolUrl?persistentId=doi:10.
↪5072/FK2/7U7YBV" \
-H "Content-Type: application/json" \
-d '{"preview": false, "locale": "en"}'
```

The JSON request body accepts the following optional parameters:

- `preview`: boolean flag to indicate if the tool should run in preview mode (default: false)
- `locale`: string specifying the locale for internationalization

The response includes:

- `toolUrl`: the URL to access the external tool
- `toolName`: the display name of the external tool
- `datasetId`: the ID of the dataset
- `preview`: whether the URL is for preview mode

Example response:

```
{
  "status": "OK",
  "data": {
    "toolUrl": "https://example.com/tool?datasetId=1234&callback=ahr...",
    "toolName": "My External Tool",
    "datasetId": 1234,
    "preview": false
  }
}
```

Get External Tool Parameters

This API call is intended as a callback that can be used by *External Tools* to retrieve signed URLs necessary for their interaction with Dataverse. It can be called directly as well.

The response is a JSON object described in the *Building External Tools* section of the API guide.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV
export VERSION=1.0
export TOOL_ID=1
```

(continues on next page)

(continued from previous page)

```
curl -H "X-Dataverse-key: $API_TOKEN" -H "Accept:application/json" "$SERVER_URL/api/
↪datasets/:persistentId/versions/$VERSION/toolparams/$TOOL_ID?persistentId=$PERSISTENT_
↪IDENTIFIER"
```

Retrieve Signposting Information

Dataverse supports *Signposting* as a discovery mechanism. Signposting involves the addition of a Link HTTP header providing summary information on GET and HEAD requests to retrieve the dataset page and a separate /linkset API call to retrieve additional information.

Signposting Link HTTP Header

Here is an example of a HTTP "Link" header from a GET or HEAD request for a dataset landing page:

```
Link: <https://doi.org/10.5072/FK2/YD5QDG>;rel="cite-as", <https://doi.org/10.
5072/FK2/YD5QDG>;rel="describedby";type="application/vnd.citationstyles.csl+json",
<https://demo.dataverse.org/api/datasets/export?exporter=OAI_ORE&persistentId=doi:10.
5072/FK2/YD5QDG>;rel="describedby";type="application/json",<https://demo.
dataverse.org/api/datasets/export?exporter=Datacite&persistentId=doi:10.5072/
FK2/YD5QDG>;rel="describedby";type="application/xml",<https://demo.dataverse.
org/api/datasets/export?exporter=oai_dc&persistentId=doi:10.5072/FK2/YD5QDG>;
rel="describedby";type="application/xml",<https://demo.dataverse.org/api/datasets/
export?exporter=oai_datacite&persistentId=doi:10.5072/FK2/YD5QDG>;rel="describedby";
type="application/xml",<https://demo.dataverse.org/api/datasets/export?exporter=schema.
org&persistentId=doi:10.5072/FK2/YD5QDG>;rel="describedby";type="application/ld+json",
<https://demo.dataverse.org/api/datasets/export?exporter=ddi&persistentId=doi:10.
5072/FK2/YD5QDG>;rel="describedby";type="application/xml",<https://demo.
dataverse.org/api/datasets/export?exporter=dcterms&persistentId=doi:10.5072/
FK2/YD5QDG>;rel="describedby";type="application/xml",<https://demo.dataverse.
org/api/datasets/export?exporter=html&persistentId=doi:10.5072/FK2/YD5QDG>;
rel="describedby";type="text/html",<https://demo.dataverse.org/api/datasets/export?
exporter=dataverse_json&persistentId=doi:10.5072/FK2/YD5QDG>;rel="describedby";
type="application/json",<https://demo.dataverse.org/api/datasets/export?
exporter=oai_ddi&persistentId=doi:10.5072/FK2/YD5QDG>;rel="describedby";
type="application/xml", <https://schema.org/AboutPage>;rel="type",<https://schema.org/
Dataset>;rel="type", <http://creativecommons.org/publicdomain/zero/1.0>;rel="license",
<https://demo.dataverse.org/api/datasets/:persistentId/versions/1.0/linkset?
persistentId=doi:10.5072/FK2/YD5QDG> ; rel="linkset";type="application/linkset+json"
```

The URL for linkset information (described below) is discoverable under the `rel="linkset";type="application/linkset+json` entry in the "Link" header, such as in the example above.

Signposting Linkset API Endpoint

The reponse includes a JSON object conforming to the Signposting specification. As part of this conformance, unlike most Dataverse API responses, the output is not wrapped in a `{"status":"OK","data":{` object. Signposting is not supported for draft dataset versions.

Like *Get Dataset Metadata Export Formats*, this API can be used to get URLs to dataset metadata export formats, but with URLs for the dataset in question.

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/YD5QDG
export VERSION=1.0

curl -H "Accept:application/json" "$SERVER_URL/api/datasets/:persistentId/versions/
↪$VERSION/linkset?persistentId=$PERSISTENT_IDENTIFIER"
```

Get Dataset By Preview URL Token

```
export SERVER_URL=https://demo.dataverse.org
export PREVIEW_URL_TOKEN=a56444bc-7697-4711-8964-e0577f055fd2

curl "$SERVER_URL/api/datasets/privateUrlDatasetVersion/$PREVIEW_URL_TOKEN"
```

If you want to include the Dataverse collections that this dataset is part of, you must set `returnOwners` query parameter to `true`.

Usage example:

```
curl "https://demo.dataverse.org/api/datasets/previewUrlDatasetVersion/a56444bc-7697-
↪4711-8964-e0577f055fd2?returnOwners=true"
```

For downloading files using a preview URL token, see the *Data Access API*.

Get Citation

This API call returns the dataset citation as seen on the dataset page, wrapped as a JSON object, with the value in the "data" sub-object's "message" key.

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/YD5QDG
export VERSION=1.0

curl -H "Accept:application/json" "$SERVER_URL/api/datasets/:persistentId/versions/
↪$VERSION/citation?persistentId=$PERSISTENT_IDENTIFIER"
```

See *Dataset Version Specifiers* for how to specify a version. By default, deaccessioned dataset versions are not included in the search when applying the :latest or :latest-published identifiers. Additionally, when filtering by a specific version tag, you will get a "not found" error if the version is deaccessioned and you do not enable the `includeDeaccessioned` option described below.

If you want to include deaccessioned dataset versions, you must set `includeDeaccessioned` query parameter to `true`.

Usage example:

```
curl -H "Accept:application/json" "$SERVER_URL/api/datasets/:persistentId/versions/
↪$VERSION/citation?persistentId=$PERSISTENT_IDENTIFIER&includeDeaccessioned=true"
```

Get Citation In Other Formats

Dataverse can also generate dataset citations in "EndNote", "RIS", "BibTeX", and "CSL" formats. Unlike the call above, which wraps the result in JSON, this API call sends the raw format with the appropriate content-type (EndNote is XML, RIS and BibTeX are plain text, and CSL is JSON). ("Internal" is also a valid value, returning the same content as the above call as HTML). This API call adds a format parameter in the API call which can be any of the values listed above.

Usage example:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/YD5QDG
export VERSION=1.0
export FORMAT=EndNote

curl "$SERVER_URL/api/datasets/:persistentId/versions/$VERSION/citation/$FORMAT?
↪persistentId=$PERSISTENT_IDENTIFIER"
```

By default, deaccessioned dataset versions are not included in the search when applying the :latest or :latest-published identifiers. Additionally, when filtering by a specific version tag, you will get a "not found" error if the version is deaccessioned and you do not enable the `includeDeaccessioned` option described below.

If you want to include deaccessioned dataset versions, you must set `includeDeaccessioned` query parameter to `true`.

Usage example:

```
curl "$SERVER_URL/api/datasets/:persistentId/versions/$VERSION/citation/$FORMAT?
↪persistentId=$PERSISTENT_IDENTIFIER&includeDeaccessioned=true"
```

The type under CSL can vary based on the dataset type, with "dataset", "software", and "review" as supported values. See also *Dataset Types*.

Note

You can also get the Datafile Citation by using the Access Datafile API. See: *Citation - Get Citation In Other Formats*.

Get Citation by Preview URL Token

```
export SERVER_URL=https://demo.dataverse.org
export PREVIEW_URL_TOKEN=a56444bc-7697-4711-8964-e0577f055fd2

curl "$SERVER_URL/api/datasets/previewUrlDatasetVersion/$PREVIEW_URL_TOKEN/citation"
```

Get Summary Field Names

See *:CustomDatasetSummaryFields* in the Installation Guide for how the list of dataset fields that summarize a dataset can be customized. Here's how to list them:

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/datasets/summaryFieldNames"
```

Get Available Dataset File Categories

This api returns a list of Categories that may be applied to the files of a given dataset.

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/YD5QDG

curl "$SERVER_URL/api/datasets/:persistentId/availableFileCategories?persistentId=
↪$PERSISTENT_IDENTIFIER"
```

Configure When a Dataset Guestbook Appears (If Enabled)

By default, users are asked to fill out a configured Guestbook when they down download files from a dataset. If enabled for a given Dataverse instance (see XYZ), users may instead be asked to fill out a Guestbook only when they request access to restricted files. This is configured by a global default, collection-level settings, or directly at the dataset level via these API calls (superuser access is required to make changes).

To see the current choice for this dataset:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/YD5QDG

curl "$SERVER_URL/api/datasets/:persistentId/guestbookEntryAtRequest?persistentId=
↪$PERSISTENT_IDENTIFIER"


The response will be true (guestbook displays when making a request), false (guestbook␣
↪displays at download), or will indicate that the dataset inherits one of these␣
↪settings.
```

To set the behavior for this dataset:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/YD5QDG

curl -X PUT -H "X-Dataverse-key:$API_TOKEN" -H Content-type:application/json -d true "
↪$SERVER_URL/api/datasets/:persistentId/guestbookEntryAtRequest?persistentId=
↪$PERSISTENT_IDENTIFIER"


This example uses true to set the behavior to guestbook at request. Note that this call␣
↪will return a 403/Forbidden response if guestbook at request functionality is not␣
↪enabled for this Dataverse instance.
```

The API can also be used to reset the dataset to use the default/inherited value:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/YD5QDG

curl -X DELETE -H "X-Dataverse-key:$API_TOKEN" -H Content-type:application/json "$SERVER_
↪URL/api/datasets/:persistentId/guestbookEntryAtRequest?persistentId=$PERSISTENT_
↪IDENTIFIER"
```

Get User Permissions on a Dataset

This API call returns the permissions that the calling user has on a particular dataset.

In particular, the user permissions that this API call checks, returned as booleans, are the following:

- Can view the unpublished dataset
- Can edit the dataset
- Can publish the dataset
- Can manage the dataset permissions
- Can delete the dataset draft

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key: $API_TOKEN" -X GET "$SERVER_URL/api/datasets/$ID/
↪userPermissions"
```

Know If a User Can Download at Least One File from a Dataset Version

This API endpoint indicates if the calling user can download at least one file from a dataset version. Note that permissions based on *Institution-Wide Shibboleth Groups* are not considered.

```
export SERVER_URL=https://demo.dataverse.org
export ID=24
export VERSION=1.0

curl -H "X-Dataverse-key: $API_TOKEN" -X GET "$SERVER_URL/api/datasets/$ID/versions/
↪$VERSION/canDownloadAtLeastOneFile"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/datasets/24/versions/1.0/canDownloadAtLeastOneFile"
```

Configure The PID Generator a Dataset Uses (If Enabled)

Dataverse can be configured to use multiple PID Providers (see the *Persistent Identifiers and Publishing Datasets* section for more information). When there are multiple PID Providers and File PIDs are enabled, it is possible to set which provider will be used to generate (mint) those PIDs. While it usually makes sense to use the same PID Provider that manages the dataset PID, there are cases, specifically if the PID Provider for the dataset PID cannot generate other PIDs with the same authority/shoulder, etc. as in the dataset PID, where another Provider is needed. Dataverse has a set of API calls to see what PID provider will be used to generate datafile PIDs and, as a superuser, to change it (to a new one or back to a default).

To see the current choice for this dataset:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/YD5QDG

curl "$SERVER_URL/api/datasets/:persistentId/pidGenerator?persistentId=$PERSISTENT_
↪IDENTIFIER"
```

The response will be the id of the PID Provider that will be used. Details of that provider's configration can be obtained via the *Get Information about Configured PID Providers*.

To set the behavior for this dataset:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/YD5QDG
export GENERATOR_ID=perma1

curl -X PUT -H "X-Dataverse-key:$API_TOKEN" -H Content-type:application/json -d
↪$GENERATOR_ID "$SERVER_URL/api/datasets/:persistentId/pidGenerator?persistentId=
↪$PERSISTENT_IDENTIFIER"
```

The PID Provider id used must be one of the those configured - see *Get Information about Configured PID Providers*. The return status code may be 200/OK, 401/403 if an api key is not sent or the user is not a superuser, or 404 if the dataset or PID provider are not found. Note that using a PIDProvider that generates DEPENDENT datafile PIDs that doesn't share the dataset PID's protocol/authority/separator/shoulder is not supported. (INDEPENDENT should be used in this case see the *Persistent Identifiers and Publishing Datasets* section for more information).

The API can also be used to reset the dataset to use the default/inherited value:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/YD5QDG

curl -X DELETE -H "X-Dataverse-key:$API_TOKEN" -H Content-type:application/json "$SERVER_
↪URL/api/datasets/:persistentId/pidGenerator?persistentId=$PERSISTENT_IDENTIFIER"
```

The default will always be the same provider as for the dataset PID if that provider can generate new PIDs, and will be the PID Provider set for the collection or the global default otherwise.

Reconcile the PID of a Dataset (If Multiple PID Providers Are Enabled)

Dataverse supports configuration with multiple Persistent Identifier (PID) providers (refer to the *Persistent Identifiers and Publishing Datasets* section for further details). This API endpoint assigns new PIDs to a draft Dataset - and, if applicable, to its Datafiles (cf. *:AllowEnablingFilePIDsPerCollection*) — using the currently configured PIDProvider. In cases where the active PIDProvider differs from the one initially used to mint the dataset's original PID, this API call facilitates reconciliation. It ensures consistency by reassigning a PID that aligns with the current provider's specifications. More specifically, for a draft dataset, a new PID is minted through the active provider, and the previously assigned PID is preserved as an alternativePersistentIdentifier. The same procedure applies to associated datafiles, provided that DataFile PIDs are enabled. (Note: If the currently configured PID provider is identical to the one originally used, this API call has no effect.)

The API is restricted to superusers and to datasets that have not already been published. (It does not make any changes to any PID Provider.) Warning: This change does not affect the storage repository, where the old PID is still used in the name of where files are stored for the dataset. If you want to remove the PID from the name used in storage, you could manually move the files offline and remove the old identifier from the database (by setting storagelocationdesignator to false for the old identifier in the alternativepersistentidentifier table). However, this step is not required for Dataverse to function correctly.

To reconcile the PID of a dataset:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/YD5QDG

curl -X PUT -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/datasets/:persistentId/
↪pidReconcile?persistentId=$PERSISTENT_IDENTIFIER"
```

Dataset Types

See *Dataset Types* in the User Guide for an overview of the feature.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

List Dataset Types

Show which dataset types are available.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/datasets/datasetTypes"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/datasetTypes"
```

Get Dataset Type

Show a dataset type by passing either its database id (e.g. "2") or its name (e.g. "software").

```
export SERVER_URL=https://demo.dataverse.org
export TYPE=software

curl $SERVER_URL/api/datasets/datasetTypes/$TYPE"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasets/datasetTypes/software"
```

Add Dataset Type

The default dataset type is "dataset" and ships with Dataverse.

Only superusers can add additional dataset types. Once added, they can only be used if a collection has been configured to allow them (see `allowedDatasetTypes` under *Change Collection Attributes*).

Here's an example of all available fields when creating a dataset type:

```
{
  "name": "foobar",
  "displayName": "Foobar",
  "description": "The best dataset type!",
  "linkedMetadataBlocks": [
    "block1",
    "block2"
  ],
  "availableLicenses": [
    "license1",
    "license2"
  ]
}
```

Here's a description of each field:

- `name` (required): Machine-readable name. Cannot be only digits.
- `displayName` (required): Human-readable name.
- `description`: A description.
- `linkedMetadataBlocks`: Linking a dataset type with one or more metadata blocks results in additional fields from those blocks appearing in the output from the *List Metadata Blocks Defined on a Dataverse Collection* API endpoint. Use the machine-readable names of the blocks. See *Link Dataset Type with Metadata Blocks* for details.
- `availableLicenses`: Limits the dataset type to certain licenses. For example, a "software" dataset type could be limited to "MIT" and "Apache-2.0". See *Set Available Licenses for a Dataset Type* for details.

Download the `datasetTypeAllFields.json` file show above, edit it to suit your needs, and use it in the following command.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org

curl -H "X-Dataverse-key:$API_TOKEN" -H "Content-Type: application/json" "$SERVER_URL/
↪api/datasets/datasetTypes" -X POST --upload-file datasetTypeAllFields.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Content-Type:␣
↪application/json" "https://demo.dataverse.org/api/datasets/datasetTypes" -X POST --
↪upload-file datasetTypeAllFields.json
```

Note that adding certain dataset types will result in a value other than "Dataset" being sent to DataCite (if you use DataCite), see *Dataset Types and DataCite* for details.

Delete Dataset Type

Superuser only. Note that if a collection has the type listed as an allowed dataset type, you will be unable to delete the dataset type until you first use the *Change Collection Attributes* to change `allowedDatasetTypes` to a dataset type (or dataset types) that you are not trying to delete.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export TYPE_ID=3

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/datasets/datasetTypes/
↪$TYPE_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/datasets/datasetTypes/3"
```

Link Dataset Type with Metadata Blocks

Linking a dataset type with one or more metadata blocks results in additional fields from those blocks appearing in the output from the *List Metadata Blocks Defined on a Dataverse Collection* API endpoint. The new frontend for Dataverse (https://github.com/IQSS/dataverse-frontend) uses the JSON output from this API endpoint to construct the page that users see when creating or editing a dataset. Once the frontend has been updated to pass in the dataset type (https://github.com/IQSS/dataverse-client-javascript/issues/210), specifying a dataset type in this way can be an alternative way to display additional metadata fields than the traditional method, which is to enable a metadata block at the collection level (see *Define Metadata Blocks for a Dataverse Collection*).

For example, a superuser could create a type called "software" and link it to the "CodeMeta" metadata block (this example is below). Then, once the new frontend allows it, the user can specify that they want to create a dataset of type software and see the additional metadata fields from the CodeMeta block when creating or editing their dataset.

This API endpoint is for superusers only.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export TYPE=software
```

(continues on next page)

(continued from previous page)

```
export JSON='["codeMeta20"]'

curl -H "X-Dataverse-key:$API_TOKEN" -H "Content-Type: application/json" "$SERVER_URL/
↪api/datasets/datasetTypes/$TYPE" -X PUT -d $JSON
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Content-Type:␣
↪application/json" "https://demo.dataverse.org/api/datasets/datasetTypes/software" -X␣
↪PUT -d '["codeMeta20"]'
```

To update the blocks that are linked, send an array with those blocks.

To remove all links to blocks, send an empty array.

Set Available Licenses for a Dataset Type

With this API a superuser may limit the available licenses for a dataset of a given type to a subset of those available in the installation.

For example, a superuser could create a type called "software" and limit the available licenses to only "MIT" and "Apache-2.0".

This API endpoint is for superusers only.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export TYPE=software
export JSON='["MIT", "Apache-2.0"]'

curl -H "X-Dataverse-key:$API_TOKEN" -H "Content-Type: application/json" "$SERVER_URL/
↪api/datasets/datasetTypes/$TYPE/licenses" -X PUT -d $JSON
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Content-Type:␣
↪application/json" "https://demo.dataverse.org/api/datasets/datasetTypes/software/
↪licenses" -X PUT -d '["MIT", "Apache-2.0"]'
```

To update the licenses available, send an array with those licenses.

To remove all links to licenses, send an empty array. If no licenses are set then the default is all active licenses in the installation.

Dataset Version Notes

Intended as *Data Provenance* information about why the version was created/how it differs from the prior version

Depositors who can edit the dataset and curators can add a version note for the draft version. Superusers can add/delete version notes for any version.

Version notes can be retrieved via the following, with authorization required to see a note on the :draft version

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=3
export VERSION=:draft

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/datasets/$ID/versions/$VERSION/
↪versionNote"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/datasets/3/versions/:draft/versionNote"
```

Notes can be set with:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=3
export VERSION=:draft
export NOTE=Files updated to correct typos

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT -d "$NOTE" "$SERVER_URL/api/datasets/$ID/
↪versions/$VERSION/versionNote"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT -d "Files updated␣
↪to correct typos" "https://demo.dataverse.org/api/datasets/3/versions/:draft/
↪versionNote"
```

And deleted via:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=3
export VERSION=2.0

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/datasets/$ID/versions/
↪$VERSION/versionNote"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/datasets/3/versions/2.0/versionNote"
```

Delete Files from a Dataset

Delete files from a dataset. This API call allows you to delete multiple files from a dataset in a single operation.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2ABCDEF

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/datasets/:persistentId/
↪deleteFiles?persistentId=$PERSISTENT_IDENTIFIER" \
-H "Content-Type: application/json" \
-d '[1, 2, 3]'
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/datasets/:persistentId/deleteFiles?persistentId=doi:10.5072/FK2ABCDEF
↪" \
-H "Content-Type: application/json" \
-d '[1, 2, 3]'
```

The JSON payload should be an array of file IDs that you want to delete from the dataset.

You must have the appropriate permissions to delete files from the dataset.

Upon success, the API will return a JSON response with a success message and the number of files deleted.

The API call will report a 400 (BAD REQUEST) error if any of the files specified do not exist or are not in the latest version of the specified dataset. The JSON payload should be an array of file IDs that you want to delete from the dataset.

Dataset Role Assignment History

Get the history of role assignments for a dataset. This API call returns a list of role assignments and revocations for the specified dataset.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=3

curl -H "X-Dataverse-key:$API_TOKEN" -H "Accept: application/json" "$SERVER_URL/api/
↪datasets/$ID/assignments/history"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Accept: application/
↪json" "https://demo.dataverse.org/api/datasets/3/assignments/history"
```

You can also use the persistent identifier instead of the numeric id:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/ABCDEF
```

(continues on next page)

(continued from previous page)

```
curl -H "X-Dataverse-key:$API_TOKEN" -H "Accept: application/json" "$SERVER_URL/api/
↪datasets/:persistentId/assignments/history?persistentId=$PERSISTENT_IDENTIFIER"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Accept: application/
↪json" "https://demo.dataverse.org/api/datasets/:persistentId/assignments/history?
↪persistentId=doi:10.5072/FK2/ABCDEF"
```

The response is a JSON array of role assignment history entries with the following structure for each entry:

```
{
  "definedOn": "3",
  "assigneeIdentifier": "@user1",
  "roleName": "Admin",
  "assignedBy": "@dataverseAdmin",
  "assignedAt": "2023-01-01T12:00:00Z",
  "revokedBy": null,
  "revokedAt": null
}
```

For revoked assignments, the "revokedBy" and "revokedAt" fields will contain values instead of null.

To retrieve the history in CSV format, change the Accept header to "text/csv":

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=3

curl -H "X-Dataverse-key:$API_TOKEN" -H "Accept: text/csv" "$SERVER_URL/api/datasets/$ID/
↪assignments/history"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Accept: text/csv"
↪"https://demo.dataverse.org/api/datasets/3/assignments/history"
```

The CSV response has column headers mirroring the JSON entries. They are internationalized (when internationalization is configured).

Note: This feature requires the *dataverse.feature.role-assignment-history* feature flag to be enabled.

Dataset Files Role Assignment History

Get the history of role assignments for the files in a dataset. This API call returns a list of role assignments and revocations for all files in the specified dataset.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=3

curl -H "X-Dataverse-key:$API_TOKEN" -H "Accept: application/json" "$SERVER_URL/api/
↪datasets/$ID/files/assignments/history"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Accept: application/
↪json" "https://demo.dataverse.org/api/datasets/3/files/assignments/history"
```

You can also use the persistent identifier instead of the numeric id:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/ABCDEF

curl -H "X-Dataverse-key:$API_TOKEN" -H "Accept: application/json" "$SERVER_URL/api/
↪datasets/:persistentId/files/assignments/history?persistentId=$PERSISTENT_IDENTIFIER"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Accept: application/
↪json" "https://demo.dataverse.org/api/datasets/:persistentId/files/assignments/history?
↪persistentId=doi:10.5072/FK2/ABCDEF"
```

The JSON response for this call is the same as for the /api/datasets/{id}/assignments/history call above with the exception that definedOn will be a comma separated list of one or more file ids.

To retrieve the history in CSV format, change the Accept header to "text/csv":

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=3

curl -H "X-Dataverse-key:$API_TOKEN" -H "Accept: text/csv" "$SERVER_URL/api/datasets/
↪files/$ID/assignments/history"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Accept: text/csv"
↪"https://demo.dataverse.org/api/datasets/3/files/assignments/history"
```

The CSV response for this call is the same as for the /api/datasets/{id}/assignments/history call above with the exception that definedOn will be a comma separated list of one or more file ids.

Note: This feature requires the *dataverse.feature.role-assignment-history* feature flag to be enabled.

Update Dataset License

Updates the license of a dataset by applying it to the draft version, or by creating a draft if none exists.

The JSON representation of a license can take two forms, depending on whether you want to specify a predefined license or define custom terms of use and access.

To set a predefined license (e.g., CC BY 4.0), provide a JSON body with the license name:

```
{
  "name": "CC BY 4.0"
}
```

To define custom terms of use and access, provide a JSON body with the following properties. All fields within `customTerms` are optional, except for the `termsOfUse` field, which is required:

```
{
  "customTerms": {
      "termsOfUse": "Your terms of use",
      "confidentialityDeclaration": "Your confidentiality declaration",
      "specialPermissions": "Your special permissions",
      "restrictions": "Your restrictions",
      "citationRequirements": "Your citation requirements",
      "depositorRequirements": "Your depositor requirements",
      "conditions": "Your conditions",
      "disclaimer": "Your disclaimer"
  }
}
```

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=3
export FILE_PATH=license.json

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/datasets/$ID/license" -H
↪"Content-type:application/json" --upload-file $FILE_PATH
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/datasets/3/license" -H "Content-type:application/json" --upload-file␣
↪license.json
```

List Reviews

Datasets can have reviews. Specifically, if a *review dataset* points at (using the `itemReviewedUrl` field) the URL form of a persistent ID of a dataset (e.g. https://doi.org/10.5072/FK2/ABCDEF) that is in the same Dataverse installation as the review dataset, the review dataset will be included in the list of reviews for the dataset. It is considered a local review. If additional "rubric" metadata blocks are enabled (see *Configure a Collection for Review Datasets*) the "metadataBlock name" must start with `rubric_` for the fields to be included in the output of this API endpoint.

An API token is optional if the review dataset has been published.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/ABCDEF

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/datasets/:persistentId/reviews?
↪persistentId=$PERSISTENT_IDENTIFIER"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/datasets/:persistentId/reviews?persistentId=doi:10.5072/FK2/ABCDEF"
```

`list-reviews.json` contains sample output of how the API response might look.

5.6.3 Files

Get JSON Representation of a File

> **Note**
>
> When a file has been assigned a persistent identifier, it can be used in the API. This is done by passing the constant `:persistentId` where the numeric id of the file is expected, and then passing the actual persistent id as a query parameter with the name `persistentId`.

This endpoint returns the file metadata present in the latest dataset version.

Example: Getting the file whose DOI is *10.5072/FK2/J8SJZB*:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/:persistentId/?persistentId=
↪$PERSISTENT_IDENTIFIER"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/:persistentId/?persistentId=doi:10.5072/FK2/J8SJZB"
```

You may get its draft version of an unpublished file if you pass an api token with view draft permissions:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER/api/files/:persistentId/?persistentId=
↪$PERSISTENT_IDENTIFIER"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/:persistentId/?persistentId=doi:10.5072/FK2/J8SJZB"
```

Show the file whose id is passed:

```
export SERVER_URL=https://demo.dataverse.org
export ID=408730

curl "$SERVER_URL/api/file/$ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/files/408730"
```

You may get its draft version of an published file if you pass an api token with view draft permissions and use the draft path parameter:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER/api/files/:persistentId/draft/?
↪persistentId=$PERSISTENT_IDENTIFIER"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/:persistentId/draft/?persistentId=doi:10.5072/FK2/J8SJZB"
```

The file id can be extracted from the response retrieved from the API which uses the persistent identifier (`/api/datasets/:persistentId/?persistentId=$PERSISTENT_IDENTIFIER`).

By default, files from deaccessioned dataset versions are not included in the search. If no accessible dataset draft version exists, the search of the latest published file will ignore dataset deaccessioned versions unless `includeDeaccessioned` query parameter is set to `true`.

Usage example:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/:persistentId/?persistentId=
↪$PERSISTENT_IDENTIFIER&includeDeaccessioned=true"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/:persistentId/?persistentId=doi:10.5072/FK2/J8SJZB&
↪includeDeaccessioned=true"
```

If you want to include the dataset version of the file in the response, there is an optional parameter for this called `returnDatasetVersion` whose default value is `false`.

Usage example:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/:persistentId/?persistentId=
↪$PERSISTENT_IDENTIFIER&returnDatasetVersion=true"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/:persistentId/?persistentId=doi:10.5072/FK2/J8SJZB&
↪returnDatasetVersion=true"
```

Get JSON Representation of a File given a Dataset Version

> **Note**
>
> When a file has been assigned a persistent identifier, it can be used in the API. This is done by passing the constant `:persistentId` where the numeric id of the file is expected, and then passing the actual persistent id as a query parameter with the name `persistentId`.

This endpoint returns the file metadata present in the requested dataset version. To specify the dataset version, you can use `:latest-published`, or `:latest`, or `:draft` or `1.0` or any other style listed under *Dataset Version Specifiers*.

Example: Getting the file whose DOI is *10.5072/FK2/J8SJZB* present in the published dataset version `1.0`:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB
export DATASET_VERSION=1.0
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/:persistentId/versions/
↪$DATASET_VERSION?persistentId=$PERSISTENT_IDENTIFIER"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/:persistentId/versions/1.0?persistentId=doi:10.5072/FK2/J8SJZB"
```

You may obtain a not found error depending on whether or not the specified version exists or you have permission to view it.

By default, files from deaccessioned dataset versions are not included in the search unless `includeDeaccessioned` query parameter is set to `true`.

Usage example:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB
export DATASET_VERSION=:latest-published
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/:persistentId/versions/
↪$DATASET_VERSION?persistentId=$PERSISTENT_IDENTIFIER&includeDeaccessioned=true"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/:persistentId/versions/:latest-published?persistentId=doi:10.5072/FK2/
↪J8SJZB&includeDeaccessioned=true"
```

If you want to include the dataset version of the file in the response, there is an optional parameter for this called `returnDatasetVersion` whose default value is `false`.

Usage example:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB
```

(continues on next page)

(continued from previous page)

```
export DATASET_VERSION=:draft
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/:persistentId/versions/
↪$DATASET_VERSION?persistentId=$PERSISTENT_IDENTIFIER&returnDatasetVersion=true"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/:persistentId/versions/:draft?persistentId=doi:10.5072/FK2/J8SJZB&
↪returnDatasetVersion=true"
```

If you want to include the dataset and collections that the file is part of in the response, there is an optional parameter for this called `returnOwners` whose default value is `false`.

Usage example:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB
export DATASET_VERSION=:draft
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/:persistentId/versions/
↪$DATASET_VERSION?persistentId=$PERSISTENT_IDENTIFIER&returnOwners=true"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/:persistentId/versions/:draft?persistentId=doi:10.5072/FK2/J8SJZB&
↪returnOwners=true"
```

Get JSON Representation of a file's versions

Gets a list of versions of a data file showing any changes that affected the file with each version. The fileIdOrPersistentId can be either "persistentId": "doi:10.5072/FK2/ADMYJF" or "datafileId": 19.

Usage example:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=1234
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB

curl -H "X-Dataverse-key: $API_TOKEN" -X GET "$SERVER_URL/api/files/$ID/
↪versionDifferences"
curl -H "X-Dataverse-key: $API_TOKEN" -X GET "$SERVER_URL/api/files/:persistentId/
↪versionDifferences?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/files/1234/versionDifferences"
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
```

(continues on next page)

(continued from previous page)

```
↪dataverse.org/api/files/:persistentId/versionDifferences?persistentId=doi:10.5072/FK2/
↪J8SJZB"
```

You can control pagination of the results using the following optional query parameters.

- `limit`: The maximum number of version differences to return.
- `offset`: The number of version differences to skip from the beginning of the list. Used for retrieving subsequent pages of results.

To aid in pagination the JSON response also includes the total number of rows (totalCount) available.

For example, to get the second page of results, with 2 items per page, you would use `limit=2` and `offset=2` (skipping the first two results).

```
curl -H "X-Dataverse-key: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/files/1234/versionDifferences?limit=2&offset=2"
```

Adding Files

Note

Files can be added via the native API but the operation is performed on the parent object, which is a dataset. Please see the *Datasets* endpoint above for more information.

Accessing (downloading) files

Note

Access API has its own section in the Guide: *Data Access API*

Note Data Access API calls can now be made using persistent identifiers (in addition to database ids). This is done by passing the constant `:persistentId` where the numeric id of the file is expected, and then passing the actual persistent id as a query parameter with the name `persistentId`.

Example: Getting the file whose DOI is *10.5072/FK2/J8SJZB*

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/J8SJZB

curl "$SERVER_URL/api/access/datafile/:persistentId/?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/access/datafile/:persistentId/?persistentId=doi:10.
↪5072/FK2/J8SJZB"
```

Note: you can use the combination of cURL's `-J` (`--remote-header-name`) and `-O` (`--remote-name`) options to save the file in its original file name, such as

```
curl -J -O "https://demo.dataverse.org/api/access/datafile/:persistentId/?
↪persistentId=doi:10.5072/FK2/J8SJZB"
```

Restrict Files

Restrict or unrestrict an existing file where `id` is the database id of the file or `pid` is the persistent id (DOI or Handle) of the file to restrict. Note that some Dataverse installations do not allow the ability to restrict files (see *:PublicInstall*). Restricting or Unrestricting a file, not in a draft version of the Dataset, will result in a new Draft version being created. Optionally the API can receive a JSON string with additional parameters related to the ability to request access to the file and the terms of that access.

A curl example using an `id`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT -d true "$SERVER_URL/api/files/$ID/restrict"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT -d true "https://
↪demo.dataverse.org/api/files/24/restrict"
```

A curl example using a `pid`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT -d true "$SERVER_URL/api/files/:persistentId/
↪restrict?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT -d true "https://
↪demo.dataverse.org/api/files/:persistentId/restrict?persistentId=doi:10.5072/FK2/AAA000
↪"
```

Optional JSON string with additional attributes:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/files/$ID/restrict" \
-H "Content-Type: application/json" \
-d '{"restrict": true, "enableAccessRequest":false, "termsOfAccess": "Reason for the␣
↪restricted access"}'
```

Note the behavior of the optional parameters:

- If restrict is false then enableAccessRequest and termsOfAccess are ignored

- If restrict is true and enableAccessRequest is false then termsOfAccess is required. A status of CONFLICT (409) will be returned if the termsOfAccess is missing

The enableAccessRequest and termsOfAccess are applied to the Draft version of the Dataset and affect all of the restricted files in said Draft version.

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/files/:persistentId/restrict?persistentId=doi:10.5072/FK2/AAA000" \
-H "Content-Type: application/json" \
-d '{"restrict": true, "enableAccessRequest":false, "termsOfAccess": "Reason for the␣
↪restricted access"}'
```

Uningest a File

Reverse the tabular data ingest process performed on a file where `ID` is the database id or `PERSISTENT_ID` is the persistent id (DOI or Handle) of the file to process.

Note that this requires "superuser" credentials to undo a successful ingest and remove the variable-level metadata and .tab version of the file. It can also be used by a user who can publish the dataset to clear the error from an unsuccessful ingest.

A curl example using an `ID`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/files/$ID/uningest"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/files/24/uningest"
```

A curl example using a `PERSISTENT_ID`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/files/:persistentId/
↪uningest?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/files/:persistentId/uningest?persistentId=doi:10.5072/FK2/AAA000"
```

Reingest a File

Attempt to ingest an existing datafile as tabular data. This API can be used on a file that was not ingested as tabular back when it was uploaded. For example, a Stata v.14 file that was uploaded before ingest support for Stata 14 was added (in Dataverse Software v.4.9). It can also be used on a file that failed to ingest due to a bug in the ingest plugin that has since been fixed (hence the name "reingest").

Note that this requires "superuser" credentials.

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/files/$ID/reingest"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/files/24/reingest"
```

A curl example using a `PERSISTENT_ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/files/:persistentId/
↪reingest?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/files/:persistentId/reingest?persistentId=doi:10.5072/FK2/AAA000"
```

Note: at present, the API cannot be used on a file that's already successfully ingested as tabular.

Redetect File Type

The Dataverse Software uses a variety of methods for determining file types (MIME types or content types) and these methods (listed below) are updated periodically. If you have files that have an unknown file type, you can have the Dataverse Software attempt to redetect the file type.

When using the curl command below, you can pass `dryRun=true` if you don't want any changes to be saved to the database. Change this to `dryRun=false` (or omit it) to save the change.

A curl example using an `id`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/files/$ID/redetect?
↪dryRun=true"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/files/24/redetect?dryRun=true"
```

A curl example using a `pid`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/files/:persistentId/
↪redetect?persistentId=$PERSISTENT_ID&dryRun=true"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/files/:persistentId/redetect?persistentId=doi:10.5072/FK2/AAA000&
↪dryRun=true"
```

Currently the following methods are used to detect file types:

- The file type detected by the browser (or sent via API).
- Custom code that reads the first few bytes. As explained at *Features that are Disabled if S3 Direct Upload is Enabled*, most of these methods are not utilized during direct upload to S3, since by nature of direct upload Dataverse never sees the contents of the file. However, this code is utilized when the "redetect" API is used.
- JHOVE: https://jhove.openpreservation.org . Note that the same applies about direct upload to S3 and the "redetect" API.
- The file extension (e.g. ".ipybn") is used, defined in a file called `MimeTypeDetectionByFileExtension.properties`.
- The file name (e.g. "Dockerfile") is used, defined in a file called `MimeTypeDetectionByFileName.properties`.

Extract NcML

As explained in the *NetCDF and HDF5* section of the User Guide, when those file types are uploaded, an attempt is made to extract an NcML file from them and store it as an auxiliary file.

This happens automatically but superusers can also manually trigger this NcML extraction process with the API endpoint below.

Note that "true" will be returned if an NcML file was created. "false" will be returned if there was an error or if the NcML file already exists (check server.log for details).

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/files/$ID/extractNcml"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/files/24/extractNcml"
```

A curl example using a PID:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/files/:persistentId/
↪extractNcml?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/files/:persistentId/extractNcml?persistentId=doi:10.5072/FK2/AAA000"
```

Replacing Files

Replace an existing file where `ID` is the database id of the file to replace or `PERSISTENT_ID` is the persistent id (DOI or Handle) of the file. Requires the `file` to be passed as well as a `jsonString` expressing the new metadata. Note that metadata such as description, directoryLabel (File Path) and tags are not carried over from the file being replaced.

Note that when a Dataverse installation is configured to use S3 storage with direct upload enabled, there is API support to send a replacement file directly to S3. This is more complex and is described in the *Direct DataFile Upload/Replace API* guide.

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X POST -F 'file=@file.extension' -F 'jsonData=
↪{json}' "$SERVER_URL/api/files/$ID/replace"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST -F 'file=@data.tsv
↪' \
  -F 'jsonData={"description":"My description.","categories":["Data"],"forceReplace
↪":false}' \
  "https://demo.dataverse.org/api/files/24/replace"
```

A curl example using a `PERSISTENT_ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl -H "X-Dataverse-key:$API_TOKEN" -X POST -F 'file=@file.extension' -F 'jsonData=
↪{json}' \
  "$SERVER_URL/api/files/:persistentId/replace?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST -F 'file=@data.tsv
↪' \
  -F 'jsonData={"description":"My description.","categories":["Data"],"forceReplace
↪":false}' \
  "https://demo.dataverse.org/api/files/:persistentId/replace?persistentId=doi:10.5072/
↪FK2/AAA000"
```

Deleting Files

Delete an existing file where `ID` is the database id of the file to delete or `PERSISTENT_ID` is the persistent id (DOI or Handle, if it exists) of the file.

Note that the behavior of deleting files depends on if the dataset has ever been published or not.

- If the dataset has never been published, the file will be deleted forever.
- If the dataset has published, the file is deleted from the draft (and future published versions).
- If the dataset has published, the deleted file can still be downloaded because it was part of a published version.

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/files/$ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/files/24"
```

A curl example using a `PERSISTENT_ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/files/:persistentId?
↪persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/files/:persistentId?persistentId=doi:10.5072/FK2/AAA000"
```

Getting File Metadata

Provides a json representation of the file metadata for an existing file where `ID` is the database id of the file to get metadata from or `PERSISTENT_ID` is the persistent id (DOI or Handle) of the file.

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl "$SERVER_URL/api/files/$ID/metadata"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/files/24/metadata"
```

A curl example using a `PERSISTENT_ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl "$SERVER_URL/api/files/:persistentId/metadata?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/files/:persistentId/metadata?persistentId=doi:10.
↪5072/FK2/AAA000"
```

The current draft can also be viewed if you have permissions and pass your API token

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/$ID/metadata/draft"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/24/metadata/draft"
```

A curl example using a `PERSISTENT_ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/:persistentId/metadata/draft?
↪persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/:persistentId/metadata/draft?persistentId=doi:10.5072/FK2/AAA000"
```

Note: The `id` returned in the json response is the id of the file metadata version.

Getting File Data Tables

This endpoint is oriented toward tabular files and provides a JSON representation of the file data tables for an existing tabular file. `ID` is the database id of the file to get the data tables from or `PERSISTENT_ID` is the persistent id (DOI or Handle) of the file.

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl $SERVER_URL/api/files/$ID/dataTables
```

The fully expanded example above (without environment variables) looks like this:

```
curl https://demo.dataverse.org/api/files/24/dataTables
```

A curl example using a `PERSISTENT_ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl "$SERVER_URL/api/files/:persistentId/dataTables?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/files/:persistentId/dataTables?persistentId=doi:10.
↪5072/FK2/AAA000"
```

Note that if the requested file is not tabular, the endpoint will return an error.

Getting File Download Count

Provides the download count for a particular file, where `ID` is the database id of the file to get the download count from or `PERSISTENT_ID` is the persistent id (DOI or Handle) of the file.

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/files/$ID/downloadCount"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/files/24/downloadCount"
```

A curl example using a PERSISTENT_ID

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/files/:persistentId/
↪downloadCount?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/files/:persistentId/downloadCount?persistentId=doi:10.5072/FK2/AAA000
↪"
```

If you are interested in download counts for multiple files, see *Metrics API*.

File Has Been Deleted

Know if a particular file that existed in a previous version of the dataset no longer exists in the latest version.

A curl example using an ID

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/files/$ID/hasBeenDeleted"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/files/24/hasBeenDeleted"
```

A curl example using a PERSISTENT_ID

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/files/:persistentId/
↪hasBeenDeleted?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET  "https://demo.
↪dataverse.org/api/files/:persistentId/hasBeenDeleted?persistentId=doi:10.5072/FK2/
↪AAA000"
```

Updating File Metadata

Updates the file metadata for an existing file where `ID` is the database id of the file to update or `PERSISTENT_ID` is the persistent id (DOI or Handle) of the file. Requires a `jsonString` expressing the new metadata. No metadata from the previous version of this file will be persisted, so if you want to update a specific field first get the json with the above command and alter the fields you want.

An optional parameter, `sourceLastUpdateTime`, can be used to verify that the file metadata being edited has not been changed since you last retrieved it, thereby avoiding potential inconsistencies. In the intended API workflow this will be the time stamp in `lastUpdateTime` from the last `GET /api/files/<id>` API call. Dataverse APIs will always report these time stamps in UTC, ISO 8601-formatted (`yyyy-MM-dd'T'HH:mm:ss'Z'`; for example: `2026-04-22T14:30:00Z`), regardless of the actual time zone used by the server. This is the only format this API will accept.

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X POST \
  -F 'jsonData={"description":"My description bbb.","provFreeform":"Test prov freeform",
↪"categories":["Data"],"dataFileTags":["Survey"],"restrict":false}' \
  "$SERVER_URL/api/files/$ID/metadata"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST \
  -F 'jsonData={"description":"My description bbb.","provFreeform":"Test prov freeform",
↪"categories":["Data"],"dataFileTags":["Survey"],"restrict":false}' \
  "https://demo.dataverse.org/api/files/24/metadata"
```

A curl example using a `PERSISTENT_ID` and the sourceLastUpdateTime parameter:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000
export UPDATE_TIME=2025-04-25T13:58:28Z

curl -H "X-Dataverse-key:$API_TOKEN" -X POST \
  -F 'jsonData={"description":"My description bbb.","provFreeform":"Test prov freeform",
↪"categories":["Data"],"dataFileTags":["Survey"],"restrict":false}' \
  "$SERVER_URL/api/files/:persistentId/metadata?persistentId=$PERSISTENT_ID&
↪sourceLastUpdateTime=$UPDATE_TIME"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST \
  -F 'jsonData={"description":"My description bbb.","provFreeform":"Test prov freeform",
↪"categories":["Data"],"dataFileTags":["Survey"],"restrict":false}' \
  "https://demo.dataverse.org/api/files/:persistentId/metadata?persistentId=doi:10.5072/
↪FK2/AAA000&sourceLastUpdateTime=2025-04-25T13:58:28Z"
```

Note: To update the 'tabularTags' property of file metadata, use the 'dataFileTags' key when making API requests. This property is used to update the 'tabularTags' of the file metadata.

Also note that dataFileTags are not versioned and changes to these will update the published version of the file.

Updating File Metadata Categories

Updates the categories for an existing file where `ID` is the database id of the file to update or `PERSISTENT_ID` is the persistent id (DOI or Handle) of the file. Requires a `jsonString` expressing the category names.

The list of "categories" will be added to the existing list unless the optional `replace=true` query parameter is included. The inclusion of this parameter will cause the pre-existing categories to be deleted and the new "categories" to be added. Sending an empty list will remove all of the pre-existing categories.

Although updating categories can also be done with the previous endpoint, this has been created to be more practical when it is only necessary to update categories and not other metadata fields.

The JSON representation of file categories (`categories.json`) looks like this:

```
{
  "categories": [
    "Data",
    "Custom"
  ]
}
```

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24
export FILE_PATH=categories.json

curl -H "X-Dataverse-key:$API_TOKEN" -X POST \
  "$SERVER_URL/api/files/$ID/metadata/categories" \
  -H "Content-type:application/json" --upload-file $FILE_PATH
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST \
  "http://demo.dataverse.org/api/files/24/metadata/categories" \
  -H "Content-type:application/json" --upload-file categories.json
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST \
  "http://demo.dataverse.org/api/files/24/metadata/categories?replace=true" \
  -H "Content-type:application/json" --upload-file categories.json
```

A curl example using a `PERSISTENT_ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000
export FILE_PATH=categories.json

curl -H "X-Dataverse-key:$API_TOKEN" -X POST \
  "$SERVER_URL/api/files/:persistentId/metadata/categories?persistentId=$PERSISTENT_ID" \
  -H "Content-type:application/json" --upload-file $FILE_PATH
curl -H "X-Dataverse-key:$API_TOKEN" -X POST \
  "$SERVER_URL/api/files/:persistentId/metadata/categories?persistentId=$PERSISTENT_ID&
```

(continues on next page)

(continued from previous page)

```
↪replace=true" \
  -H "Content-type:application/json" --upload-file $FILE_PATH
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST \
  "https://demo.dataverse.org/api/files/:persistentId/metadata/categories?
↪persistentId=doi:10.5072/FK2/AAA000" \
  -H "Content-type:application/json" --upload-file categories.json
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST \
  "https://demo.dataverse.org/api/files/:persistentId/metadata/categories?
↪persistentId=doi:10.5072/FK2/AAA000&replace=true" \
  -H "Content-type:application/json" --upload-file tags.json
```

Note that if the specified categories do not exist, they will be created.

Updating File Tabular Tags

Updates the tabular tags for an existing tabular file where `ID` is the database id of the file to update or `PERSISTENT_ID` is the persistent id (DOI or Handle) of the file. Requires a `jsonString` expressing the tabular tag names.

The list of "tabularTags" will be added to the existing list unless the optional `replace=true` query parameter is included. The inclusion of this parameter will cause the pre-existing tags to be deleted and the "tabularTags" to be added. Sending an empty list will remove all of the pre-existing tags.

The JSON representation of tabular tags (`tags.json`) looks like this:

```
{
  "tabularTags": [
    "Survey",
    "Genomics"
  ]
}
```

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24
export FILE_PATH=tags.json

curl -H "X-Dataverse-key:$API_TOKEN" -X POST \
  "$SERVER_URL/api/files/$ID/metadata/tabularTags" \
  -H "Content-type:application/json" --upload-file $FILE_PATH
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST \
  "http://demo.dataverse.org/api/files/24/metadata/tabularTags" \
  -H "Content-type:application/json" --upload-file tags.json
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST \
  "http://demo.dataverse.org/api/files/24/metadata/tabularTags?replace=true" \
  -H "Content-type:application/json" --upload-file tags.json
```

A curl example using a PERSISTENT_ID

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000
export FILE_PATH=tags.json

curl -H "X-Dataverse-key:$API_TOKEN" -X POST \
  "$SERVER_URL/api/files/:persistentId/metadata/tabularTags?persistentId=$PERSISTENT_ID"␣
↪\
  -H "Content-type:application/json" --upload-file $FILE_PATH
curl -H "X-Dataverse-key:$API_TOKEN" -X POST \
  "$SERVER_URL/api/files/:persistentId/metadata/tabularTags?persistentId=$PERSISTENT_ID&
↪replace=true" \
  -H "Content-type:application/json" --upload-file $FILE_PATH
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST \
  "https://demo.dataverse.org/api/files/:persistentId/metadata/tabularTags?
↪persistentId=doi:10.5072/FK2/AAA000" \
  -H "Content-type:application/json" --upload-file tags.json
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST \
  "https://demo.dataverse.org/api/files/:persistentId/metadata/tabularTags?
↪persistentId=doi:10.5072/FK2/AAA000&replace=true" \
  -H "Content-type:application/json" --upload-file tags.json
```

Note that the specified tabular tags must be valid. The supported tags are:

- Survey
- Time Series
- Panel
- Event
- Genomics
- Network
- Geospatial

Editing Variable Level Metadata

Updates variable level metadata using ddi xml FILE, where ID is file id.

A curl example using an ID

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24
export FILE=dct.xml

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/edit/$ID" --upload-file
↪$FILE
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/edit/24" --upload-file dct.xml
```

You can download `dct.xml` from the example above to see what the XML looks like.

Get File Citation as JSON

This API is for getting the file citation as it appears on the file landing page. It is formatted in HTML and encoded in JSON.

To specify the version, you can use `:latest-published` or `:draft` or `1.0` or any other style listed under *Dataset Version Specifiers*.

When the dataset version is published, authentication is not required:

```
export SERVER_URL=https://demo.dataverse.org
export FILE_ID=42
export DATASET_VERSION=:latest-published

curl "$SERVER_URL/api/files/$FILE_ID/versions/$DATASET_VERSION/citation"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/files/42/versions/:latest-published/citation"
```

When the dataset version is a draft or deaccessioned, authentication is required.

By default, deaccessioned dataset versions are not included in the search when applying the :latest or :latest-published identifiers. Additionally, when filtering by a specific version tag, you will get a "unauthorized" error if the version is deaccessioned and you do not enable the `includeDeaccessioned` option described below.

If you want to include deaccessioned dataset versions, you must set `includeDeaccessioned` query parameter to `true`.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export FILE_ID=42
export DATASET_VERSION=:draft
export INCLUDE_DEACCESSIONED=true

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/$FILE_ID/versions/$DATASET_
↪VERSION/citation?includeDeaccessioned=$INCLUDE_DEACCESSIONED"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/42/versions/:draft/citation?includeDeaccessioned=true"
```

If your file has a persistent identifier (PID, such as a DOI), you can pass it using the technique described under *Get JSON Representation of a File*.

This API is not for downloading various citation formats such as EndNote XML, RIS, or BibTeX. This functionality has been requested in https://github.com/IQSS/dataverse/issues/3140 and https://github.com/IQSS/dataverse/issues/9994.

Provenance

Get Provenance JSON for an uploaded file

A curl example using an ID

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/$ID/prov-json"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/24/prov-json"
```

A curl example using a PERSISTENT_ID

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/:persistentId/prov-json?
↪persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/:persistentId/prov-json?persistentId=doi:10.5072/FK2/AAA000"
```

Get Provenance Description for an uploaded file

A curl example using an ID

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/$ID/prov-freeform"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/24/prov-freeform"
```

A curl example using a PERSISTENT_ID

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000
```

(continues on next page)

(continued from previous page)

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/files/:persistentId/prov-freeform?
↪persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/files/:persistentId/prov-freeform?persistentId=doi:10.5072/FK2/AAA000"
```

Create/Update Provenance JSON and provide related entity name for an uploaded file

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24
export ENTITY_NAME="..."
export FILE_PATH=provenance.json

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/files/$ID/prov-json?
↪entityName=$ENTITY_NAME" -H "Content-type:application/json" --upload-file $FILE_PATH
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/files/24/prov-json?entityName=..." -H "Content-type:application/json
↪" --upload-file provenance.json
```

A curl example using a `PERSISTENT_ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000
export ENTITY_NAME="..."
export FILE_PATH=provenance.json

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/files/:persistentId/prov-
↪json?persistentId=$PERSISTENT_ID&entityName=$ENTITY_NAME" -H "Content-type:application/
↪json" --upload-file $FILE_PATH
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/files/:persistentId/prov-json?persistentId=doi:10.5072/FK2/AAA000&
↪entityName=..." -H "Content-type:application/json" --upload-file provenance.json
```

Create/Update Provenance Description for an uploaded file

Requires a JSON file with the description connected to a key named "text"

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24
export FILE_PATH=provenance.json

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/files/$ID/prov-freeform" -
↪H "Content-type:application/json" --upload-file $FILE_PATH
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/files/24/prov-freeform" -H "Content-type:application/json" --upload-
↪file provenance.json
```

A curl example using a `PERSISTENT_ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000
export FILE_PATH=provenance.json

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/files/:persistentId/prov-
↪freeform?persistentId=$PERSISTENT_ID" -H "Content-type:application/json" --upload-file
↪$FILE_PATH
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/files/:persistentId/prov-freeform?persistentId=doi:10.5072/FK2/AAA000
↪" -H "Content-type:application/json" --upload-file provenance.json
```

See a sample JSON file `file-provenance.json` from https://openprovenance.org (c.f. Huynh, Trung Dong and Moreau, Luc (2014) ProvStore: a public provenance repository. At 5th International Provenance and Annotation Workshop (IPAW'14), Cologne, Germany, 09-13 Jun 2014. pp. 275-277).

Delete Provenance JSON for an uploaded file

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/files/$ID/prov-json"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/files/24/prov-json"
```

A curl example using a PERSISTENT_ID

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_ID=doi:10.5072/FK2/AAA000

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/files/:persistentId/prov-
↪json?persistentId=$PERSISTENT_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/files/:persistentId/prov-json?persistentId=doi:10.5072/FK2/AAA000"
```

Datafile Integrity

Starting with the release 4.10 the size of the saved original file (for an ingested tabular datafile) is stored in the database. The following API will retrieve and permanently store the sizes for any already existing saved originals:

```
export SERVER_URL=https://localhost

curl "$SERVER_URL/api/admin/datafiles/integrity/fixmissingoriginalsizes"
```

with limit parameter:

```
export SERVER_URL=https://localhost
export LIMIT=10

curl "$SERVER_URL/api/admin/datafiles/integrity/fixmissingoriginalsizes?limit=$LIMIT"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://localhost/api/admin/datafiles/integrity/fixmissingoriginalsizes"
```

with limit parameter:

```
curl "https://localhost/api/admin/datafiles/integrity/fixmissingoriginalsizes?limit=10"
```

Note the optional "limit" parameter. Without it, the API will attempt to populate the sizes for all the saved originals that don't have them in the database yet. Otherwise it will do so for the first N such datafiles.

By default, the admin API calls are blocked and can only be called from localhost. See more details in *:BlockedApiEndpoints* and *:BlockedApiPolicy* settings in *Configuration*.

Get File External Tool URL

This API call generates a URL for accessing an external tool (see *External Tools*) that operates at the file level. The URL includes necessary authentication tokens and parameters based on the user's permissions and the tool's configuration.

Authentication is required for draft, restricted, embargoed, or expired (retention period) files; the user must have appropriate permissions.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export FILE_ID=42
export TOOL_ID=3

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/files/$FILE_ID/
↪externalTool/$TOOL_ID/toolUrl" \
-H "Content-Type: application/json" \
-d '{"preview": false, "locale": "en"}'
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/files/42/externalTool/3/toolUrl" \
-H "Content-Type: application/json" \
-d '{"preview": false, "locale": "en"}'
```

The JSON request body accepts the following optional parameters:

- `preview`: boolean flag to indicate if the tool should run in preview mode (default: false)
- `locale`: string specifying the locale for internationalization

The response includes:

- `toolUrl`: the URL to access the external tool
- `toolName`: the display name of the external tool
- `fileId`: the ID of the file
- `preview`: whether the URL is for preview mode

Example response:

```
{
  "status": "OK",
  "data": {
    "toolUrl": "https://example.com/tool?fileId=42&callback=ahr...",
    "toolName": "File Viewer Tool",
    "fileId": 42,
    "preview": false
  }
}
```

Get File External Tool Parameters

This API call is intended as a callback that can be used by *External Tools* to retrieve signed URLs necessary for their interaction with Dataverse files. It can be called directly as well.

The response is a JSON object described in the *Building External Tools* section of the API guide.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export FILE_ID=42
export FILE_METADATA_ID=123
export TOOL_ID=3

curl -H "X-Dataverse-key: $API_TOKEN" -H "Accept:application/json" "$SERVER_URL/api/
↪files/$FILE_ID/metadata/$FILE_METADATA_ID/toolparams/$TOOL_ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -H "Accept:application/
↪json" "https://demo.dataverse.org/api/files/42/metadata/123/toolparams/3"
```

Get External Tool Parameters

This API call is intended as a callback that can be used by *External Tools* to retrieve signed Urls necessary for their interaction with Dataverse. It can be called directly as well. (Note that the required FILEMETADATA_ID is the "id" returned in the JSON response from the /api/files/$FILE_ID/metadata call.)

The response is a JSON object described in the *Building External Tools* section of the API guide.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export FILE_ID=3
export FILEMETADATA_ID=1
export TOOL_ID=1

curl -H "X-Dataverse-key: $API_TOKEN" -H "Accept:application/json" "$SERVER_URL/api/
↪files/$FILE_ID/metadata/$FILEMETADATA_ID/toolparams/$TOOL_ID"
```

Get Fixity Algorithm

This API call can be used to discover the configured fixity/checksum algorithm being used by a Dataverse installation (as configured by - *:FileFixityChecksumAlgorithm*). Currently, the possible values are MD5, SHA-1, SHA-256, and SHA-512. This algorithm will be used when the Dataverse software manages a file upload and should be used by external clients uploading files to a Dataverse instance. (Existing files may or may not have checksums with this algorithm.)

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/files/fixityAlgorithm"
```

5.6.4 Users Token Management

The following endpoints will allow users to manage their API tokens.

Find a Token's Expiration Date

In order to obtain the expiration date of a token use:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/users/token"
```

Recreate a Token

In order to obtain a new token use:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/users/token/recreate"
```

This endpoint by default will return a response message indicating the user identifier and the new token.

To also include the expiration time in the response message, the query parameter `returnExpiration` must be set to true:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/users/token/recreate?
↪returnExpiration=true"
```

Delete a Token

In order to delete a token use:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/users/token"
```

5.6.5 Builtin Users

Builtin users are known as "Username/Email and Password" users in the *Account Creation + Management* of the User Guide. The Dataverse installation stores a password (encrypted, of course) for these users, which differs from "remote" users such as Shibboleth or OAuth users where the password is stored elsewhere. See also *Auth Modes: Local vs. Remote vs. Both* section of Configuration in the Installation Guide. It's a valid configuration of a Dataverse installation to not use builtin users at all.

Create a Builtin User

For security reasons, builtin users cannot be created via API unless the team who runs the Dataverse installation has populated a database setting called `:BuiltinUsersKey`, which is described under *Securing Your Installation* and *Database Settings* sections of Configuration in the Installation Guide. You will need to know the value of `:BuiltinUsersKey` before you can proceed.

To create a builtin user via API, you must first construct a JSON document. You can download `user-add.json` or copy the text below as a starting point and edit as necessary.

```
{
  "firstName": "Lisa",
  "lastName": "Simpson",
  "userName": "lsimpson",
  "affiliation": "Springfield",
  "position": "Student",
  "email": "lsimpson@mailinator.com"
}
```

Place this `user-add.json` file in your current directory and run the following curl command, substituting variables as necessary. Note that both the password of the new user and the value of `:BuiltinUsersKey` are passed as query parameters:

```
curl -d @user-add.json -H "Content-type:application/json" "$SERVER_URL/api/builtin-users?
↪password=$NEWUSER_PASSWORD&key=$BUILTIN_USERS_KEY"
```

Optionally, you may use a third query parameter "sendEmailNotification=false" to explicitly disable sending an email notification to the new user.

5.6.6 Roles

A role is a set of permissions.

JSON Representation of a Role

The JSON representation of a role (`roles.json`) looks like this:

```
{
  "alias": "sys1",
  "name": "Restricted System Role",
  "description": "A person who may only add datasets.",
  "permissions": [
    "AddDataset"
  ]
}
```

Note

alias is constrained to a length of 16 characters

Create Role

Roles can be created globally (*Create Global Role*) or for individual Dataverse collections (*Create a New Role in a Dataverse Collection*).

Show Role

You must have `ManageDataversePermissions` to be able to show a role that was created using *Create a New Role in a Dataverse Collection*. Global roles (*Create Global Role*) can only be shown with a superuser API token.

An example using a role alias:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ALIAS=sys1

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/roles/:alias?alias=$ALIAS"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/roles/:alias?alias=sys1"
```

An example using a role id:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=11

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/roles/$ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/roles/11"
```

Delete Role

An example using a role id:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/roles/$ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/roles/24"
```

An example using a role alias:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ALIAS=sys1

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/roles/:alias?alias=$ALIAS
↪"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/roles/:alias?alias=sys1"
```

Get User Selectable Roles

Returns the appropriate roles that the calling user can use as filters when searching within their data.

This endpoint returns the same set of roles displayed as filters on the MyData UI page. The logic for determining which roles are returned is as follows:

- If the user is a superuser, all available roles are returned.
- If the user is not a superuser, only their assigned roles are returned.
- If the user has no assigned roles, all roles are returned as a fallback.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/roles/userSelectable"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  "https://demo.dataverse.
↪org/api/roles/userSelectable"
```

5.6.7 Explicit Groups

Create New Explicit Group

Explicit groups list their members explicitly. These groups are defined in Dataverse collections, which is why their API endpoint is under `api/dataverses/$id/`, where `$id` is the id of the Dataverse collection.

Create a new explicit group under Dataverse collection `$id`:

```
POST http://$server/api/dataverses/$id/groups
```

Data being POSTed is json-formatted description of the group:

```
{
 "description":"Describe the group here",
 "displayName":"Close Collaborators",
 "aliasInOwner":"ccs"
}
```

A curl example:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/dataverses/$ID/groups" --
```

(continues on next page)

(continued from previous page)

```
↪data '{"description":"Describe the group here","displayName":"Close Collaborators",
↪"aliasInOwner":"ccs"}'
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  "https://demo.dataverse.
↪org/api/dataverses/24/groups" --data '{"description":"Describe the group here",
↪"displayName":"Close Collaborators", "aliasInOwner":"ccs"}'
```

List Explicit Groups in a Dataverse Collection

List explicit groups under Dataverse collection `ID`. A curl example using an `ID`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$ID/groups"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  "https://demo.dataverse.
↪org/api/dataverses/24/groups"
```

Show Single Group in a Dataverse Collection

Show group `$GROUP_ALIAS` under dataverse `$DATAVERSE_ID` and a `$GROUP_ALIAS`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export GROUP_ALIAS=ccs
export DATAVERSE_ID=24

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/dataverses/$DATAVERSE_ID/groups/
↪$GROUP_ALIAS"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"  "https://demo.dataverse.
↪org/api/dataverses/24/groups/ccs"
```

Update Group in a Dataverse Collection

Show group `$GROUP_ALIAS` under dataverse `$DATAVERSE_ID` and a `$GROUP_ALIAS`. The request body is the same as the create group one, except that the group alias cannot be changed. Thus, the field `aliasInOwner` is ignored.:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export GROUP_ALIAS=ccs
export DATAVERSE_ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/dataverses/$DATAVERSE_ID/
↪groups/$GROUP_ALIAS" --data '{"description":"Describe the group here","displayName":
↪"Close Collaborators"}'
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/dataverses/24/groups/ccs" --data '{"description":"Describe the group␣
↪here","displayName":"Close Collaborators"}'
```

Delete Group from a Dataverse Collection

Delete group `$GROUP_ALIAS` under Dataverse collection `$DATAVERSE_ID`:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export GROUP_ALIAS=ccs
export DATAVERSE_ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/dataverses/$DATAVERSE_ID/
↪groups/$GROUP_ALIAS"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/dataverses/24/groups/ccs"
```

Add Multiple Role Assignees to an Explicit Group

Bulk add role assignees to an explicit group. The request body is a JSON array of role assignee identifiers, such as `@admin`, `&ip/localhosts` or `:authenticated-users`:

```
POST http://$server/api/dataverses/$dv/groups/$groupAlias/roleAssignees
```

Add a Role Assignee to an Explicit Group

Add a single role assignee to a group. Request body is ignored:

```
PUT http://$server/api/dataverses/$dv/groups/$groupAlias/roleAssignees/
↪$roleAssigneeIdentifier
```

Remove a Role Assignee from an Explicit Group

Remove a single role assignee from an explicit group:

```
DELETE http://$server/api/dataverses/$dv/groups/$groupAlias/roleAssignees/
↪$roleAssigneeIdentifier
```

5.6.8 Shibboleth Groups

Management of Shibboleth groups via API is documented in the *Shibboleth* section of the Installation Guide.

5.6.9 Info

Show Dataverse Software Version and Build Number

Get the Dataverse installation version. The response contains the version and build numbers:

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/info/version"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/info/version"
```

Show Dataverse Installation Server Name

Get the server name. This is useful when a Dataverse installation is composed of multiple app servers behind a load balancer:

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/info/server"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/info/server"
```

Show Custom Popup Text for Publishing Datasets

For now, only the value for the *:DatasetPublishPopupCustomText* setting from the Configuration section of the Installation Guide is exposed:

> **Note**
>
> See *Show Disclaimer for Publishing Datasets* if you want the user to acknowledge before publishing.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/info/settings/:DatasetPublishPopupCustomText"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/info/settings/:DatasetPublishPopupCustomText"
```

Show Custom Popup Text for Submitting Datasets For Review

For now, only the value for the *:DatasetSubmitForReviewPopupCustomText* setting from the Configuration section of the Installation Guide is exposed:

> **Note**
>
> See *Show Disclaimer for Submitting For Review Datasets* if you want the user to acknowledge before submitting for review.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/info/settings/:DatasetSubmitForReviewPopupCustomText"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/info/settings/:DatasetSubmitForReviewPopupCustomText
↪"
```

Show Disclaimer for Publishing Datasets

The setting *:PublishDatasetDisclaimerText*, when set, will prevent a draft dataset from being published through the UI without the user acknowledging the disclaimer.

> **Note**
>
> See *Show Custom Popup Text for Publishing Datasets* if the user acknowledgment is not required but you want the message to be displayed in the UI.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/info/settings/:PublishDatasetDisclaimerText"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/info/settings/:PublishDatasetDisclaimerText"
```

Show Disclaimer for Submitting For Review Datasets

The setting *:SubmitForReviewDatasetDisclaimerText*, when set, will prevent a draft dataset from being submitted for review through the UI without the user acknowledging the disclaimer.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/info/settings/:SubmitForReviewDatasetDisclaimerText"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/info/settings/:SubmitForReviewDatasetDisclaimerText"
```

Get Application Terms of Use (General Terms of Use)

In the UI, Application Terms of Use is called "General Terms of Use" and can be seen when you sign up for an account. The terms come from the database setting *:ApplicationTermsOfUse*. If you have enabled *Internationalization* you can pass a two-character language code (e.g. "en") as the `lang` parameter.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org
export LANG=en

curl "$SERVER_URL/api/info/applicationTermsOfUse?lang=$LANG"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/info/applicationTermsOfUse?lang=en"
```

Get API Terms of Use URL

Get API Terms of Use. The response contains the text value inserted as API Terms of use which uses the database setting `:ApiTermsOfUse`:

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/info/apiTermsOfUse"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/info/apiTermsOfUse"
```

Show Support Of Incomplete Metadata Deposition

Learn if an instance has been configured to allow deposition of incomplete datasets via the API. See also *Create a Dataset in a Dataverse Collection* and *dataverse.api.allow-incomplete-metadata*

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/info/settings/incompleteMetadataViaApi"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/info/settings/incompleteMetadataViaApi"
```

Get Zip File Download Limit

Get the configured zip file download limit. The response contains the long value of the limit in bytes.

This limit comes from the database setting *:ZipDownloadLimit* if set, or the default value if the database setting is not set, which is 104857600 (100MB).

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/info/zipDownloadLimit"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/info/zipDownloadLimit"
```

Get Maximum Embargo Duration In Months

Get the maximum embargo duration in months, if available, configured through the database setting *:MaxEmbargoDurationInMonths* from the Configuration section of the Installation Guide.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/info/settings/:MaxEmbargoDurationInMonths"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/info/settings/:MaxEmbargoDurationInMonths"
```

Get Dataset Metadata Export Formats

Get the available dataset metadata export formats, including formats from external exporters (see *Available Dataset Metadata Exporters*).

The response contains a JSON object with the available format names as keys (these can be passed to *Export Metadata of a Dataset in Various Formats*), and values as objects with the following properties:

- `displayName`
- `mediaType`
- `isHarvestable`
- `isVisibleInUserInterface` (corresponds to isAvailableToUsers)
- `XMLNameSpace` (only for XML exporters)
- `XMLSchemaLocation` (only for XML exporters)
- `XMLSchemaVersion` (only for XML exporters)

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/info/exportFormats"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/info/exportFormats"
```

Get Customization File Contents

The Customization API is used to retrieve the analytics-code.html as well as other customization file contents.

See also *Web Analytics Code* in the Configuration section of the Installation Guide and *Branding Your Installation*

The Content-Type returned in the header is based on the media type of the file being returned (example: analytics-code.html returns "text/html; charset=UTF-8"

Valid types are "homePage", "header", "footer", "style", "analytics", and "logo".

A curl example getting the analytics-code

```
export SERVER_URL=https://demo.dataverse.org
export TYPE=analytics

curl -X GET "$SERVER_URL/api/info/settings/customization/$TYPE"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -X GET "https://demo.dataverse.org/api/info/settings/customization/analytics"
```

5.6.10 Metadata Blocks

See also *Exploring Metadata Blocks*.

Show Info About All Metadata Blocks

Lists brief info about all metadata blocks registered in the system.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/metadatablocks"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/metadatablocks"
```

This endpoint supports the following optional query parameters:

- `returnDatasetFieldTypes`: Whether or not to return the dataset field types present in each metadata block. If not set, the default value is false.
- `onlyDisplayedOnCreate`: Whether or not to return only the metadata blocks that are displayed on dataset creation. If `returnDatasetFieldTypes` is true, only the dataset field types shown on dataset creation will be returned within each metadata block. If not set, the default value is false.

An example using the optional query parameters is presented below:

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/metadatablocks?returnDatasetFieldTypes=true&
↪onlyDisplayedOnCreate=true"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/metadatablocks?returnDatasetFieldTypes=true&
↪onlyDisplayedOnCreate=true"
```

Show Info About Single Metadata Block

Return data about the block whose `identifier` is passed, including allowed controlled vocabulary values. `identifier` can either be the block's database id, or its name (i.e. "citation").

```
export SERVER_URL=https://demo.dataverse.org
export IDENTIFIER=citation

curl "$SERVER_URL/api/metadatablocks/$IDENTIFIER"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/metadatablocks/citation"
```

5.6.11 Dataset Fields

List All Facetable Dataset Fields

List all facetable dataset fields defined in the installation.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/datasetfields/facetables"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/datasetfields/facetables"
```

Set displayOnCreate for a Dataset Field

Set displayOnCreate for a dataset field. See also *Metadata Customization* in the Admin Guide.

```
export SERVER_URL=http://localhost:8080
export FIELD=subtitle
export BOOLEAN=true

curl -X POST "$SERVER_URL/api/admin/datasetfield/setDisplayOnCreate?datasetFieldType=
↪$FIELD&setDisplayOnCreate=$BOOLEAN"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -X POST "http://localhost:8080/api/admin/datasetfield/setDisplayOnCreate?
↪datasetFieldType=studyAssayCellType&setDisplayOnCreate=true"
```

5.6.12 Notifications

See *Notifications* in the User Guide for an overview. For a list of all the notification types mentioned below (e.g. ASSIGNROLE), see *Letting Users Manage Notifications* in the Admin Guide.

Get All Notifications by User

Each user can get a dump of their notifications by passing in their API token:

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/notifications/all"
```

The expected OK (200) response looks something like this:

```
{
    "status": "OK",
    "totalCount": 15,
    "data": [
      {
          "id": 38,
          "type": "CREATEACC",
          "displayAsRead": true,
```

(continues on next page)

(continued from previous page)

```
            "subjectText": "Root: Your account has been created",
            "messageText": "Hello, \nWelcome to...",
            "sentTimestamp": "2025-07-21T19:15:37Z"
        }
...
```

This endpoint supports several optional query parameters to filter and paginate the results.

The `inAppNotificationFormat` parameter, if sent as `true`, retrieves the fields needed to build the in-app notifications for the Notifications section of the Dataverse UI, omitting fields related to email notifications.

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/notifications/all?
↪inAppNotificationFormat=true"
```

The `onlyUnread` parameter, if sent as `true`, filters the results to include only notifications that have not been marked as read.

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/notifications/all?onlyUnread=true"
```

The `limit` and `offset` parameters can be used for pagination. `limit` specifies the maximum number of notifications to return, and `offset` specifies the number of notifications to skip from the beginning of the list. For example, to retrieve notifications 11 through 15:

To aid in pagination the JSON response also includes the total number of rows (totalCount) available.

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/notifications/all?limit=5&offset=10
↪"
```

All parameters can be combined. For instance, to get the first page of 10 unread notifications in the in-app format:

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/notifications/all?
↪inAppNotificationFormat=true&onlyUnread=true&limit=1&offset=0"
```

The expected OK (200) response for an in-app format request looks something like this:

```
{
    "status": "OK",
    "totalCount": 15,
    "data": [
      {
          "id": 79,
          "type": "CREATEACC",
          "displayAsRead": false,
          "sentTimestamp": "2025-08-08T08:00:16Z",
          "installationBrandName": "Your Installation Name",
          "userGuidesBaseUrl": "https://guides.dataverse.org",
          "userGuidesVersion": "6.7.1",
          "userGuidesSectionPath": "user/index.html"
      }
    ]
}
...
```

Get Unread Count

You can get a count of your unread notifications as shown below.

```
curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/notifications/unreadCount"
```

Mark Notification As Read

After finding the ID of a notification using *Get All Notifications by User*, you can pass it to the "markAsRead" API endpoint as shown below. Note that this endpoint is idempotent; you can mark an already-read notification as read over and over.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export NOTIFICATION_ID=555

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/notifications/$NOTIFICATION_
↪ID/markAsRead"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/notifications/555/markAsRead"
```

Delete Notification by User

Each user can delete notifications by passing in their API token and specifying notification ID (e.g., 555):

```
export NOTIFICATION_ID=555

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/notifications/
↪$NOTIFICATION_ID"
```

Get All Muted In-app Notifications by User

Each user can get a list of their muted in-app notification types by passing in their API token:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/notifications/
↪mutedNotifications"
```

Mute In-app Notification by User

Each user can mute in-app notifications by passing in their API token and specifying notification type to be muted (e.g., ASSIGNROLE):

```
export NOTIFICATION_TYPE=ASSIGNROLE

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/notifications/
↪mutedNotifications/$NOTIFICATION_TYPE"
```

Unmute In-app Notification by User

Each user can unmute in-app notifications by passing in their API token and specifying notification type to be unmuted (e.g., ASSIGNROLE):

```
export NOTIFICATION_TYPE=ASSIGNROLE

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/notifications/
↪mutedNotifications/$NOTIFICATION_TYPE"
```

Get All Muted Email Notifications by User

Each user can get a list of their muted email notification types by passing in their API token:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/notifications/mutedEmails"
```

Mute Email Notification by User

Each user can mute email notifications by passing in their API token and specifying notification type to be muted (e.g., ASSIGNROLE):

```
export NOTIFICATION_TYPE=ASSIGNROLE

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/notifications/mutedEmails/
↪$NOTIFICATION_TYPE"
```

Unmute Email Notification by User

Each user can unmute email notifications by passing in their API token and specifying notification type to be unmuted (e.g., ASSIGNROLE):

```
export NOTIFICATION_TYPE=ASSIGNROLE

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/notifications/
↪mutedEmails/$NOTIFICATION_TYPE"
```

5.6.13 User Information

Get User Information in JSON Format

Each user can get a dump of their basic information in JSON format by passing in their API token:

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/users/:me"
```

5.6.14 Managing Harvesting Server and Sets

This API can be used to manage the Harvesting sets that your installation makes available over OAI-PMH. For more information, see the *Managing Harvesting Server and Sets* section of the Admin Guide.

List All Harvesting Sets

Shows all Harvesting Sets defined in the installation:

```
GET http://$SERVER/api/harvest/server/oaisets/
```

List A Specific Harvesting Set

Shows a Harvesting Set with a defined specname:

```
GET http://$SERVER/api/harvest/server/oaisets/$specname
```

Create a Harvesting Set

To create a harvesting set you must supply a JSON file that contains the following fields:

- Name: Alpha-numeric may also contain -, _, or %, but no spaces. It must also be unique in the installation.
- Definition: A search query to select the datasets to be harvested. For example, a query containing authorName:YYY would include all datasets where 'YYY' is the authorName.
- Description: Text that describes the harvesting set. The description appears in the Manage Harvesting Sets dashboard and in API responses. This field is optional.

An example JSON file would look like this:

```
{
 "name":"ffAuthor",
 "definition":"authorName:Finch, Fiona",
 "description":"Fiona Finch's Datasets"
}
```

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/harvest/server/oaisets/add
↪" --upload-file harvestset-finch.json
```

The fully expanded example above (without the environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/harvest/server/oaisets/add" --upload-file "harvestset-finch.json"
```

Only users with superuser permissions may create harvesting sets.

Modify an Existing Harvesting Set

To modify a harvesting set, you must supply a JSON file that contains one or both of the following fields:

- Definition: A search query to select the datasets to be harvested. For example, a query containing authorName:YYY would include all datasets where 'YYY' is the authorName.
- Description: Text that describes the harvesting set. The description appears in the Manage Harvesting Sets dashboard and in API responses. This field is optional.

Note that you may not modify the name of an existing harvesting set.

An example JSON file would look like this:

```
{
 "definition":"authorName:Finch, Fiona AND subject:trees",
 "description":"Fiona Finch's Datasets with subject of trees"
}
```

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export SPECNAME=ffAuthor

curl -H "X-Dataverse-key:$API_TOKEN" -X PUT "$SERVER_URL/api/harvest/server/oaisets/
↪$SPECNAME" --upload-file modify-harvestset-finch.json
```

The fully expanded example above (without the environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X PUT "https://demo.
↪dataverse.org/api/harvest/server/oaisets/ffAuthor" --upload-file "modify-harvestset-
↪finch.json"
```

Only users with superuser permissions may modify harvesting sets.

Delete an Existing Harvesting Set

To delete a harvesting set, use the set's database name. For example, to delete an existing harvesting set whose database name is "ffAuthor":

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export SPECNAME=ffAuthor

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/harvest/server/oaisets/
↪$SPECNAME"
```

The fully expanded example above (without the environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/harvest/server/oaisets/ffAuthor"
```

Only users with superuser permissions may delete harvesting sets.

5.6.15 Managing Harvesting Clients

The following API can be used to create and manage "Harvesting Clients". A Harvesting Client is a configuration entry that allows your Dataverse installation to harvest and index metadata from a specific remote location, either regularly, on a configured schedule, or on a one-off basis. For more information, see the *Managing Harvesting Clients* section of the Admin Guide.

List All Configured Harvesting Clients

Shows all the harvesting clients configured.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/harvest/clients"
```

The fully expanded example above (without the environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/harvest/clients"
```

Show a Specific Harvesting Client

Shows a harvesting client by nickname.

```
export SERVER_URL=https://demo.dataverse.org
export NICKNAME=myclient

curl "$SERVER_URL/api/harvest/clients/$NICKNAME"
```

The fully expanded example above (without the environment variables) looks like this:

```
curl "https://demo.dataverse.org/api/harvest/clients/myclient"
```

The output will look something like the following.

```
{
  "status":"OK",
  {
    "data": {
      "lastDatasetsFailed": "22",
      "lastDatasetsDeleted": "0",
```

(continues on next page)

(continued from previous page)

```
    "metadataFormat": "oai_dc",
    "archiveDescription": "This Dataset is harvested from our partners. Clicking the␣
↪link will take you directly to the archival source of the data.",
    "archiveUrl": "https://dataverse.foo.edu",
    "harvestUrl": "https://dataverse.foo.edu/oai",
    "style": "dataverse",
    "type": "oai",
    "dataverseAlias": "fooData",
    "nickName": "myClient",
    "sourceName": "",
    "set": "fooSet",
    "useOaiIdentifiersAsPids": false,
    "useListRecords": false,
    "schedule": "none",
    "status": "inActive",
    "lastHarvest": "Thu Oct 13 14:48:57 EDT 2022",
    "lastResult": "SUCCESS",
    "lastSuccessful": "Thu Oct 13 14:48:57 EDT 2022",
    "lastNonEmpty": "Thu Oct 13 14:48:57 EDT 2022",
    "lastDatasetsHarvested": "137"
  }
}
```

Create a Harvesting Client

To create a harvesting client you must supply a JSON file that describes the configuration, similarly to the output of the GET API above. The following fields are mandatory:

- `dataverseAlias`: The alias of an existing collection where harvested datasets will be deposited
- `harvestUrl`: The URL of the remote OAI archive
- `archiveUrl`: The URL of the remote archive that will be used in the redirect links pointing back to the archival locations of the harvested records. It may or may not be on the same server as the harvestUrl above. If this OAI archive is another Dataverse installation, it will be the same URL as harvestUrl minus the "/oai". For example: https://demo.dataverse.org/ vs. https://demo.dataverse.org/oai
- `metadataFormat`: A supported metadata format. As of writing this the supported formats are "oai_dc", "oai_ddi" and "dataverse_json".

The following optional fields are supported:

- `sourceName`: When the *dataverse.feature.index-harvested-metadata-source* feature flag is enabled, sourceName will override the nickname in the Metadata Source facet. It can be used to group the content from many harvesting clients under the same name.
- `archiveDescription`: What the name suggests. If not supplied, will default to "This Dataset is harvested from our partners. Clicking the link will take you directly to the archival source of the data."
- `set`: The OAI set on the remote server. If not supplied, will default to none, i.e., "harvest everything". (Note: see the note below on using sets when harvesting from DataCite; this is new as of v6.6).
- `style`: Defaults to "default" - a generic OAI archive. (Make sure to use "dataverse" when configuring harvesting from another Dataverse installation).

- `schedule`: Defaults to "none" (not scheduled). Two formats are supported, for weekly- and daily-scheduled harvests; examples: `Weekly, Sat 5 AM`; `Daily, 11 PM`. Note that if a schedule definition is not formatted exactly as described here, it will be ignored silently and the client will be left unscheduled.
- `customHeaders`: This can be used to configure this client with a specific HTTP header that will be added to every OAI request. This is to accommodate a use case where the remote server requires this header to supply some form of a token in order to offer some content not available to other clients. See the example below. Multiple headers can be supplied separated by *\n* - actual "backslash" and "n" characters, not a single "new line" character.
- `allowHarvestingMissingCVV`: Flag to allow datasets to be harvested with Controlled Vocabulary Values that existed in the originating Dataverse Project but are not in the harvesting Dataverse Project. (Default is false). Currently only settable using API.
- `useOaiIdentifiersAsPids`: Defaults to false; if set to true, the harvester will attempt to use the identifier from the OAI-PMH record header as the **first choice** for the persistent id of the harvested dataset. When set to false, Dataverse will still attempt to use this identifier, but only if none of the `<dc:identifier>` entries in the OAI_DC record contain a valid persistent id (this is new as of v6.5).
- `useListRecords`: Defaults to false; if set to true, the harvester will attempt to retrieve multiple records in a single pass using the OAI-PMH verb ListRecords. By default, our harvester relies on the combination of ListIdentifiers followed by multiple GetRecord calls for each individual record. Note that this option is required when configuring harvesting from DataCite. (this is new as of v6.6).

Generally, the API will accept the output of the GET version of the API for an existing client as valid input, but some fields will be ignored.

You can download this `harvesting-client.json` file to use as a starting point.

```
{
  "nickName": "zenodo",
  "dataverseAlias": "zenodoHarvested",
  "harvestUrl": "https://zenodo.org/oai2d",
  "archiveUrl": "https://zenodo.org",
  "archiveDescription": "Moissonné depuis la collection LMOPS de l'entrepôt Zenodo. En␣
↪cliquant sur ce jeu de données, vous serez redirigé vers Zenodo.",
  "metadataFormat": "oai_dc",
  "customHeaders": "x-oai-api-key: xxxyyyzzz",
  "set": "user-lmops",
  "schedule": "Weekly, Sat 5 AM",
  "allowHarvestingMissingCVV":true
}
```

Something important to keep in mind about this API is that, unlike the harvesting clients GUI, it will create a client with the values supplied without making any attempts to validate them in real time. In other words, for the *harvestUrl* it will accept anything that looks like a well-formed url, without making any OAI calls to verify that the name of the set and/or the metadata format entered are supported by it. This is by design, to give an admin an option to still be able to create a client, in a rare case when it cannot be done via the GUI because of some real time failures in an exchange with an otherwise valid OAI server. This however puts the responsibility on the admin to supply the values already confirmed to be valid.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

`nickName` in the JSON file and `$NICKNAME` in the URL path below is the name identifying the new client. It should be alpha-numeric and may also contain -, _, or %, but no spaces. It must be unique in the installation.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=http://localhost:8080
export NICKNAME=zenodo

curl -H "X-Dataverse-key:$API_TOKEN" -X POST -H "Content-Type: application/json" "
↪$SERVER_URL/api/harvest/clients/$NICKNAME" --upload-file harvesting-client.json
```

The fully expanded example above (without the environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST -H "Content-Type:␣
↪application/json" "http://localhost:8080/api/harvest/clients/zenodo" --upload-file
↪"harvesting-client.json"
```

The output will look something like the following.

```
{
  "status": "OK",
  "data": {
    "metadataFormat": "oai_dc",
    "archiveDescription": "Moissonné depuis la collection LMOPS de l'entrepôt Zenodo. En␣
↪cliquant sur ce jeu de données, vous serez redirigé vers Zenodo.",
    "archiveUrl": "https://zenodo.org",
    "harvestUrl": "https://zenodo.org/oai2d",
    "style": "default",
    "type": "oai",
    "dataverseAlias": "zenodoHarvested",
    "nickName": "zenodo",
    "set": "user-lmops",
    "schedule": "none",
    "status": "inActive",
    "lastHarvest": "N/A",
    "lastSuccessful": "N/A",
    "lastNonEmpty": "N/A",
    "lastDatasetsHarvested": "N/A",
    "lastDatasetsDeleted": "N/A"
  }
}
```

Only users with superuser permissions may create or configure harvesting clients.

Modify a Harvesting Client

Similar to the API above, using the same JSON format, but run on an existing client and using the PUT method instead of POST.

Delete a Harvesting Client

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=http://localhost:8080
export NICKNAME=zenodo

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/harvest/clients/$NICKNAME
↪"
```

The fully expanded example above (without the environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "http://
↪localhost:8080/api/harvest/clients/zenodo"
```

Only users with superuser permissions may delete harvesting clients.

Harvesting from DataCite

The following 2 options are **required** when harvesting from DataCite (https://oai.datacite.org/oai):

```
"useOaiIdentifiersAsPids": true,
"useListRecords": true,
```

There are two ways the `set` parameter can be used when harvesting from DataCite:

- DataCite maintains pre-configured OAI sets for every subscribing institution that registers DOIs with them. This can be used to harvest the entire set of metadata registered by this organization or school, etc. (this is identical to how the set parameter is used with any other standard OAI archive);
- As a unique, proprietary DataCite feature, it can be used to harvest virtually any arbitrary subset of records (potentially spanning different institutions and authorities, etc.). Any query that the DataCite search API understands can be used as an OAI set name (!). For example, the following search query finds one specific dataset:

```
https://api.datacite.org/dois?query=doi:10.7910/DVN/TJCLKP
```

you can now create a single-record OAI set by using its base64-encoded form as the set name:

```
echo "doi:10.7910/DVN/TJCLKP" | base64
ZG9pOjEwLjc5MTAvRFZOL1RKQ0xLUAo=
```

use the encoded string above prefixed by the ~ character in your harvesting client configuration:

```
"set": "~ZG9pOjEwLjc5MTAvRFZOL1RKQ0xLUAo="
```

The following configuration will create a client that will harvest the IQSS dataset specified above on a weekly schedule:

```
{
  "useOaiIdentifiersAsPids": true,
  "useListRecords": true,
```

(continues on next page)

(continued from previous page)

```
  "set": "~ZG9pOjEwLjc5MTAvRFZOL1RKQ0xLUAo=",
  "nickName": "iqssTJCLKP",
  "dataverseAlias": "harvestedCollection",
  "type": "oai",
  "style": "default",
  "harvestUrl": "https://oai.datacite.org/oai",
  "archiveUrl": "https://oai.datacite.org",
  "archiveDescription": "The metadata for this IQSS Dataset was harvested from DataCite.␣
↪Clicking the dataset link will take you directly to the original archival location, as␣
↪registered with DataCite.",
  "schedule": "Weekly, Tue 4 AM",
  "metadataFormat": "oai_dc"
}
```

The queries can be as complex and/or long as necessary, with sub-queries combined via logical ANDs and ORs. Please keep in mind that white spaces must be encoded as `%20`. For example, the following query:

```
prefix:10.17603 AND (types.resourceType:Report* OR types.resourceType:Mission*)
```

must be encoded as follows:

```
echo "prefix:10.17603%20AND%20(types.resourceType:Report*%20OR%20types.
↪resourceType:Mission*)" | base64
cHJlZml4OjEwLjE3NjAzJTIwQU5EJTIwKHR5cGVzLnJlc291cmNlVHlwZTpSZXBvcnQqJTIwT1IlMjB0eXBlcy5yZXNvdXJjZVR5cGU
```

5.6.16 PIDs

PIDs is short for Persistent IDentifiers. Examples include DOI or Handle. There are some additional PID operations listed in the *Managing Datasets and Dataverse Collections* section of the Admin Guide.

Get Info on a PID

Get information on a PID, especially its "state" such as "draft" or "findable". Currently, this API only works on DataCite DOIs. A superuser API token is required.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PID=doi:10.70122/FK2/9BXT5O

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/pids?persistentId=$PID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/pids?persistentId=doi:10.70122/FK2/9BXT5O"
```

List Unreserved PIDs

Get a list of PIDs that have not been reserved on the PID provider side. This can happen, for example, if a dataset is created while the PID provider is down. A superuser API token is required.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/pids/unreserved"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/pids/unreserved"
```

Reserve a PID

Reserve a PID for a dataset if not yet registered, and, if FilePIDs are enabled, reserve any file PIDs that are not yet registered. A superuser API token is required.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PID=doi:10.70122/FK2/9BXT5O

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/pids/:persistentId/reserve?
↪persistentId=$PID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "https://demo.
↪dataverse.org/api/pids/:persistentId/reserve?persistentId=doi:10.70122/FK2/9BXT5O"
```

Delete a PID

Delete PID (this is only possible for PIDs that are in the "draft" state) and within a Dataverse installation, set `globalidcreatetime` to null and `identifierregistered` to false. A superuser API token is required.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PID=doi:10.70122/FK2/9BXT5O

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/pids/:persistentId/
↪delete?persistentId=$PID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/pids/:persistentId/delete?persistentId=doi:10.70122/FK2/9BXT5O"
```

Get Information about Configured PID Providers

Dataverse can be configured with one or more PID Providers that it uses to create new PIDs and manage existing ones. This API call returns a JSONObject listing the configured providers and details about the protocol/authority/separator/shoulder they manage, along with information about about how new dataset and datafile PIDs are generated. See the *Persistent Identifiers and Publishing Datasets* section for more information.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/pids/providers"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/pids/providers"
```

Get the id of the PID Provider Managing a Given PID

Dataverse can be configured with one or more PID Providers that it uses to create new PIDs and manage existing ones. This API call returns the string id of the PID Provider than manages a given PID. See the *Persistent Identifiers and Publishing Datasets* section for more information. Delete PID (this is only possible for PIDs that are in the "draft" state) and within a Dataverse installation, set `globalidcreatetime` to null and `identifierregistered` to false. A superuser API token is required.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PID=doi:10.70122/FK2/9BXT5O

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/pids/providers/$PID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/pids/providers/doi:10.70122/FK2/9BXT5O"
```

If the PID is not managed by Dataverse, this call will report if the PID is recognized as a valid PID for a given protocol (doi, hdl, or perma)
or will return a 400/Bad Request response if it is not.

5.6.17 Admin

This is the administrative part of the API. For security reasons, it is absolutely essential that you block it before allowing public access to a Dataverse installation. See *Blocking API Endpoints* in the Installation Guide for details.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

Manage Database Settings

These are the API endpoints for managing the *Database Settings* listed in the Installation Guide.

List All Database Settings

```
export SERVER_URL="http://localhost:8080"

curl "$SERVER_URL/api/admin/settings"
```

The fully expanded example above (without environment variables) looks like this:

```
curl http://localhost:8080/api/admin/settings
```

Get Single Database Setting

```
export SERVER_URL="http://localhost:8080"
export NAME=":UploadMethods"

curl "$SERVER_URL/api/admin/settings/$NAME"
```

The fully expanded example above (without environment variables) looks like this:

```
curl http://localhost:8080/api/admin/settings/:UploadMethods
```

Get Single Database Setting With Language/Locale

A small number of settings, most notably *:ApplicationTermsOfUse*, can be saved in multiple languages.

Use two-character ISO 639-1 language codes.

```
export SERVER_URL="http://localhost:8080"
export NAME=":ApplicationTermsOfUse"
export LANG="en"

curl "$SERVER_URL/api/admin/settings/$NAME/lang/$LANG"
```

The fully expanded example above (without environment variables) looks like this:

```
curl http://localhost:8080/api/admin/settings/:ApplicationTermsOfUse/lang/en
```

Configure Single Database Setting

```
export SERVER_URL="http://localhost:8080"
export NAME=":InstallationName"
export VALUE="LibreScholar"

curl -X PUT "$SERVER_URL/api/admin/settings/$NAME" -d "$VALUE"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -X PUT http://localhost:8080/api/admin/settings/:InstallationName -d LibreScholar
```

Note: `NAME` values are validated for existence and compliance.

Configure Single Database Setting With Language/Locale

A small number of settings, most notably *:ApplicationTermsOfUse*, can be saved in multiple languages.

Use two-character ISO 639-1 language codes.

```
export SERVER_URL="http://localhost:8080"
export NAME=":ApplicationTermsOfUse"
export LANG="fr"

curl -X PUT "$SERVER_URL/api/admin/settings/$NAME/lang/$LANG" --upload-file /tmp/apptou_
↪fr.html
```

The fully expanded example above (without environment variables) looks like this:

```
curl -X PUT http://localhost:8080/api/admin/settings/:ApplicationTermsOfUse/lang/fr --
↪upload-file /tmp/apptou_fr.html
```

Note: `NAME` and `LANG` values are validated for existence and compliance.

Configure All Database Settings

Using a JSON file, replace all settings in a single idempotent and atomic operation and delete any settings not present in that JSON file.

Use the JSON `data` object in output of `GET /api/admin/settings` (*List All Database Settings*) for the JSON input structure for this endpoint. To put this concretely, you can save just the `data` object for your existing settings to disk by filtering them through `jq` like this:

```
curl http://localhost:8080/api/admin/settings | jq '.data' > /tmp/all-settings.json
```

Then you can use this "all-settings.json" file as a starting point for your input file. The *Configuration* page of the Installation Guide has a *complete list of all the available settings*. Note that settings in the JSON file are validated for existence and compliance.

```
export SERVER_URL="http://localhost:8080"

curl -X PUT -H "Content-type:application/json" "$SERVER_URL/api/admin/settings" --upload-
↪file /tmp/all-settings.json
```

The fully expanded example above (without environment variables) looks like this:

```
curl -X PUT -H "Content-type:application/json" http://localhost:8080/api/admin/settings
↪-upload-file /tmp/all-settings.json
```

Delete Single Database Setting

```
export SERVER_URL="http://localhost:8080"
export NAME=":InstallationName"

curl -X DELETE "$SERVER_URL/api/admin/settings/$NAME"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -X DELETE http://localhost:8080/api/admin/settings/:InstallationName
```

Delete Single Database Setting With Language/Locale

A small number of settings, most notably *:ApplicationTermsOfUse*, can be saved in multiple languages.

Use two-character ISO 639-1 language codes.

```
export SERVER_URL="http://localhost:8080"
export NAME=":ApplicationTermsOfUse"
export LANG="fr"

curl -X DELETE "$SERVER_URL/api/admin/settings/$NAME/lang/$LANG"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -X DELETE http://localhost:8080/api/admin/settings/:ApplicationTermsOfUse/lang/fr
```

List All Feature Flags

Experimental and preview features are sometimes hidden behind feature flags. See *Feature Flags* in the Installation Guide for a list of flags and how to configure them.

This API endpoint provides a list of feature flags and "enabled" or "disabled" for each one.

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=http://localhost:8080

curl "$SERVER_URL/api/admin/featureFlags"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "http://localhost:8080/api/admin/featureFlags"
```

Show Feature Flag Status

This endpoint reports "enabled" as true for false for a single feature flag. (For all flags, see *List All Feature Flags*.)

> **Note**
>
> See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=http://localhost:8080
export FLAG=DATASET_TYPES

curl "$SERVER_URL/api/admin/featureFlags/$FLAG"
```

The fully expanded example above (without environment variables) looks like this:

```
curl "http://localhost:8080/api/admin/featureFlags/DATASET_TYPES"
```

Manage Banner Messages

> **Warning**
>
> Adding a banner message with a language that is not supported by the installation will result in a 500-Internal Server Error response when trying to access to the /bannerMessage.

Communications to users can be handled via banner messages that are displayed at the top of all pages within your Dataverse installation. Two types of banners can be configured:

- A banner message where dismissibleByUser is set to false will be displayed to anyone viewing the application. These messages will be dismissible for a given session but will be displayed in any subsequent session until they are deleted by the Admin. This type of banner message is useful for situations such as upcoming maintenance windows and other downtime.
- A banner message where dismissibleByUser is set to true is intended to be used in situations where the Admin wants to make sure that all logged in users see a certain notification. These banner messages will only be displayed to users when they are logged in and can be dismissed by the logged in user. Once they have been dismissed by a user, that user will not see the message again. This type of banner message is useful for situations where a message needs to communicated once, such as a minor terms of use update or an update about a new workflow in your Dataverse installation.

Note that HTML can be included in banner messages.

Add a Banner Message:

```
curl -H "Content-type:application/json" -X POST "http://$SERVER/api/admin/bannerMessage"␣
↪--upload-file messages.json
```

Where `messages.json` looks like this:

```
{
  "dismissibleByUser": "true",
  "messageTexts": [
  {
    "lang": "en",
    "message": "Dismissible Banner Message added via API"
  },
  {
    "lang": "fr",
    "message": "Message de bannière ajouté via l'API"
  }
  ]
}
```

Get a list of active Banner Messages:

```
curl -X GET "http://$SERVER/api/admin/bannerMessage"
```

Delete a Banner Message by its id:

```
curl -X DELETE "http://$SERVER/api/admin/bannerMessage/$id"
```

Deactivate a Banner Message by its id (allows you to hide a message while retaining information about which users have dismissed the banner):

```
curl -X PUT "http://$SERVER/api/admin/bannerMessage/$id/deactivate"
```

List Authentication Provider Factories

List the authentication provider factories. The alias field of these is used while configuring the providers themselves.

```
GET http://$SERVER/api/admin/authenticationProviderFactories
```

List Authentication Providers

List all the authentication providers in the system (both enabled and disabled):

```
GET http://$SERVER/api/admin/authenticationProviders
```

Add Authentication Provider

Add new authentication provider. The POST data is in JSON format, similar to the JSON retrieved from this command's `GET` counterpart.

```
POST http://$SERVER/api/admin/authenticationProviders
```

Note

This endpoint will create providers for both JSF and SPA. Use *JVM Options* / *MicroProfile Config* if you need to create SPA only providers.

Show Authentication Provider

Show data about an authentication provider:

```
GET http://$SERVER/api/admin/authenticationProviders/$id
```

Enable or Disable an Authentication Provider

Enable or disable an authentication provider (denoted by `id`):

```
PUT http://$SERVER/api/admin/authenticationProviders/$id/enabled
```

> **Note**
>
> The former endpoint, ending with `:enabled` (that is, with a colon), is still supported, but deprecated.

Check If an Authentication Provider is Enabled

Check whether an authentication proider is enabled:

```
GET http://$SERVER/api/admin/authenticationProviders/$id/enabled
```

The body of the request should be either `true` or `false`. Content type has to be `application/json`, like so:

```
curl -H "Content-type: application/json"  -X POST -d"false" "http://localhost:8080/api/
↪admin/authenticationProviders/echo-dignified/:enabled"
```

Delete an Authentication Provider

Deletes an authentication provider from the system. The command succeeds even if there is no such provider, as the postcondition holds: there is no provider by that id after the command returns.

```
DELETE http://$SERVER/api/admin/authenticationProviders/$id/
```

List Global Roles

List all global roles in the system.

```
GET http://$SERVER/api/admin/roles
```

Create Global Role

Creates a global role in the Dataverse installation. The data POSTed are assumed to be a role JSON.

```
POST http://$SERVER/api/admin/roles
```

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=http://localhost:8080

curl -H "Content-Type: application/json" -H "X-Dataverse-key:$API_TOKEN" -X POST "
↪$SERVER_URL/api/admin/roles" --upload-file roles.json
```

`roles.json` see *JSON Representation of a Role*

Update Global Role

Update a global role in the Dataverse installation. The PUTed data is assumed to be a complete JSON role as it will overwrite the existing role.

```
PUT http://$SERVER/api/admin/roles/$ID
```

A curl example using an `ID`

```
export SERVER_URL=http://localhost:8080
export ID=24

curl -H "Content-Type: application/json" -X PUT "$SERVER_URL/api/admin/roles/$ID" --
↪upload-file roles.json
```

`roles.json` see *JSON Representation of a Role*

Delete Global Role

Deletes an `DataverseRole` whose `id` is passed.

```
DELETE http://$SERVER/api/admin/roles/$ID
```

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=http://localhost:8080
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/admin/roles/$ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/admin/roles/24"
```

A curl example using a Role alias `ALIAS`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ALIAS=roleAlias

curl -H "X-Dataverse-key:$API_TOKEN" -X DELETE "$SERVER_URL/api/admin/roles/:alias?alias=
↪$ALIAS"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X DELETE "https://demo.
↪dataverse.org/api/admin/roles/:alias?alias=roleAlias"
```

List Users

List users with the options to search and "page" through results. Only accessible to superusers. Optional parameters:

- `searchTerm` A string that matches the beginning of a user identifier, first name, last name or email address.
- `itemsPerPage` The number of detailed results to return. The default is 25. This number has no limit. e.g. You could set it to 1000 to return 1,000 results
- `selectedPage` The page of results to return. The default is 1.
- `sortKey` A string that represents a field that is used for sorting the results. Possible values are "id", "useridentifier" (username), "lastname" (last name), "firstname" (first name), "email" (email address), "affiliation" (affiliation), "superuser" (flag that denotes if the user is an administrator of the site), "position", "createdtime" (created time), "lastlogintime" (last login time), "lastapiusetime" (last API use time), "authproviderid" (the authentication provider ID). To sort in reverse order you can add " desc" e.g. "id desc". The default value is "useridentifier".

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/admin/list-users"

# sort by createdtime (the creation time of the account)
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/admin/list-users?
↪sortKey=createdtime"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/admin/list-users"

# sort by createdtime (the creation time of the account)
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/admin/list-users?sortKey=createdtime"
```

Sample output appears below.

- When multiple pages of results exist, the `selectedPage` parameters may be specified.
- Note, the resulting `pagination` section includes `pageCount`, `previousPageNumber`, `nextPageNumber`, and other variables that may be used to re-create the UI.

```
{
    "status":"OK",
    "data":{
        "userCount":27,
        "selectedPage":1,
        "pagination":{
            "isNecessary":true,
            "numResults":27,
            "numResultsString":"27",
            "docsPerPage":25,
            "selectedPageNumber":1,
            "pageCount":2,
            "hasPreviousPageNumber":false,
            "previousPageNumber":1,
            "hasNextPageNumber":true,
            "nextPageNumber":2,
            "startResultNumber":1,
            "endResultNumber":25,
            "startResultNumberString":"1",
            "endResultNumberString":"25",
            "remainingResults":2,
            "numberNextResults":2,
            "pageNumberList":[
                1,
                2
            ]
        },
        "bundleStrings":{
            "userId":"ID",
            "userIdentifier":"Username",
            "lastName":"Last Name ",
            "firstName":"First Name ",
            "email":"Email",
            "affiliation":"Affiliation",
            "position":"Position",
            "isSuperuser":"Superuser",
            "authenticationProvider":"Authentication",
            "roles":"Roles",
            "createdTime":"Created Time",
            "lastLoginTime":"Last Login Time",
            "lastApiUseTime":"Last API Use Time"
        },
        "users":[
            {
                "id":8,
                "userIdentifier":"created1",
                "lastName":"created1",
                "firstName":"created1",
                "email":"created1@g.com",
                "affiliation":"hello",
                "isSuperuser":false,
                "authenticationProvider":"BuiltinAuthenticationProvider",
                "roles":"Curator",
```


(continues on next page)

(continued from previous page)

```
                "createdTime":"2017-06-28 10:36:29.444"
            },
            {
                "id":9,
                "userIdentifier":"created8",
                "lastName":"created8",
                "firstName":"created8",
                "email":"created8@g.com",
                "isSuperuser":false,
                "authenticationProvider":"BuiltinAuthenticationProvider",
                "roles":"Curator",
                "createdTime":"2000-01-01 00:00:00.0"
            },
            {
                "id":1,
                "userIdentifier":"dataverseAdmin",
                "lastName":"Admin",
                "firstName":"Dataverse",
                "email":"dataverse@mailinator2.com",
                "affiliation":"Dataverse.org",
                "position":"Admin",
                "isSuperuser":true,
                "authenticationProvider":"BuiltinAuthenticationProvider",
                "roles":"Admin, Contributor",
                "createdTime":"2000-01-01 00:00:00.0",
                "lastLoginTime":"2017-07-03 12:22:35.926",
                "lastApiUseTime":"2017-07-03 12:55:57.186"
            }

            // ... 22 more user documents ...
        ]
    }
}
```

Note

"List all users" `GET http://$SERVER/api/admin/authenticatedUsers` is deprecated, but supported.

List Single User

List user whose `identifier` (without the `@` sign) is passed:

```
GET http://$SERVER/api/admin/authenticatedUsers/$identifier
```

Sample output using "dataverseAdmin" as the `identifier`:

```
{
  "authenticationProviderId": "builtin",
  "persistentUserId": "dataverseAdmin",
  "position": "Admin",
```

(continues on next page)

(continued from previous page)

```
  "id": 1,
  "identifier": "@dataverseAdmin",
  "displayName": "Dataverse Admin",
  "firstName": "Dataverse",
  "lastName": "Admin",
  "email": "dataverse@mailinator.com",
  "superuser": true,
  "affiliation": "Dataverse.org"
}
```

Create an Authenticated User

Create an authenticatedUser:

```
POST http://$SERVER/api/admin/authenticatedUsers
```

POSTed JSON example:

```
{
  "authenticationProviderId": "orcid",
  "persistentUserId": "0000-0002-3283-0661",
  "identifier": "@pete",
  "firstName": "Pete K.",
  "lastName": "Dataversky",
  "email": "pete@mailinator.com"
}
```

Merge User Accounts

If a user has created multiple accounts and has been performed actions under both accounts that need to be preserved, these accounts can be combined. One account can be merged into another account and all data associated with both accounts will be combined in the surviving account. Only accessible to superusers.:

```
POST https://$SERVER/api/users/$toMergeIdentifier/mergeIntoUser/$continuingIdentifier
```

Example: `curl -H "X-Dataverse-key: $API_TOKEN" -X POST "http://demo.dataverse.org/api/users/jsmith2/mergeIntoUser/jsmith"`

This action moves account data from jsmith2 into the account jsmith and deletes the account of jsmith2.

Note: User accounts can only be merged if they are either both active or both deactivated. See *deactivate a user*.

Change User Identifier

Changes identifier for user in `AuthenticatedUser`, `BuiltinUser`, `AuthenticatedUserLookup` & `RoleAssignment`. Allows them to log in with the new identifier. Only accessible to superusers.:

```
POST http://$SERVER/api/users/$oldIdentifier/changeIdentifier/$newIdentifier
```

Example: `curl -H "X-Dataverse-key: $API_TOKEN" -X POST "https://demo.dataverse.org/api/users/johnsmith/changeIdentifier/jsmith"`

This action changes the identifier of user johnsmith to jsmith.

Toggle Superuser Status

Toggle the superuser status of a user.

Note

This endpoint is deprecated as explained in *API Changelog (Breaking Changes)*. Please use the *Set Superuser Status* endpoint instead.

```
export SERVER_URL=http://localhost:8080
export USERNAME=jdoe
curl -X POST "$SERVER_URL/api/admin/superuser/$USERNAME"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -X POST "http://localhost:8080/api/admin/superuser/jdoe"
```

Set Superuser Status

Specify the superuser status of a user with a boolean value (`true` or `false`).

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of `export` below.

```
export SERVER_URL=http://localhost:8080
export USERNAME=jdoe
export IS_SUPERUSER=true
curl -X PUT "$SERVER_URL/api/admin/superuser/$USERNAME" -d "$IS_SUPERUSER"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -X PUT "http://localhost:8080/api/admin/superuser/jdoe" -d true
```

Delete a User

Deletes an `AuthenticatedUser` whose `identifier` (without the `@` sign) is passed.

```
DELETE http://$SERVER/api/admin/authenticatedUsers/$identifier
```

Deletes an `AuthenticatedUser` whose `id` is passed.

```
DELETE http://$SERVER/api/admin/authenticatedUsers/id/$id
```

Note: If the user has performed certain actions such as creating or contributing to a Dataset or downloading a file they cannot be deleted. To see where in the database these actions are stored you can use the *Show User Traces* API. If a user cannot be deleted for this reason, you can choose to *deactivate a user*.

Deactivate a User

Deactivates a user. A superuser API token is not required but the command will operate using the first superuser it finds.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export SERVER_URL=http://localhost:8080
export USERNAME=jdoe

curl -X POST "$SERVER_URL/api/admin/authenticatedUsers/$USERNAME/deactivate"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -X POST "http://localhost:8080/api/admin/authenticatedUsers/jdoe/deactivate"
```

The database ID of the user can be passed instead of the username.

```
export SERVER_URL=http://localhost:8080
export USERID=42

curl -X POST "$SERVER_URL/api/admin/authenticatedUsers/id/$USERID/deactivate"
```

Note: A primary purpose of most Dataverse installations is to serve an archive. In the archival space, there are best practices around the tracking of data access and the tracking of modifications to data and metadata. In support of these key workflows, a simple mechanism to delete users that have performed edit or access actions in the system is not provided. Providing a Deactivate User endpoint for users who have taken certain actions in the system alongside a Delete User endpoint to remove users that haven't taken certain actions in the system is by design.

This is an irreversible action. There is no option to undeactivate a user.

Deactivating a user with this endpoint will:

- Deactivate the user's ability to log in to the Dataverse installation. A message will be shown, stating that the account has been deactivated. The user will not able to create a new account with the same email address, ORCID, Shibboleth, or other login type.
- Deactivate the user's ability to use the API

- Remove the user's access from all Dataverse collections, datasets and files
- Prevent a user from being assigned any roles
- Cancel any pending file access requests generated by the user
- Remove the user from all groups
- No longer have notifications generated or sent by the Dataverse installation
- Prevent the account from being converted into an OAuth or Shibboleth account.
- Prevent the user from becoming a superuser.

Deactivating a user with this endpoint will keep:

- The user's contributions to datasets, including dataset creation, file uploads, and publishing.
- The user's access history to datafiles in the Dataverse installation, including guestbook records.
- The user's account information (specifically name, email, affiliation, and position)

Show User Traces

Show the traces that the user has left in the system, such as datasets created, guestbooks filled out, etc. This can be useful for understanding why a user cannot be deleted. A superuser API token is required.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export USERNAME=jdoe

curl -H "X-Dataverse-key:$API_TOKEN" -X GET "$SERVER_URL/api/users/$USERNAME/traces"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X GET "https://demo.
↪dataverse.org/api/users/jdoe/traces"
```

Remove All Roles from a User

Removes all roles from the user. This is equivalent of clicking the "Remove All Roles" button in the superuser dashboard. Note that you can preview the roles that will be removed with the *Show User Traces* API. A superuser API token is required.

Note

See *curl Examples and Environment Variables* if you are unfamiliar with the use of export below.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export USERNAME=jdoe

curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/users/$USERNAME/removeRoles
↪"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -X POST "http://
↪localhost:8080/api/users/jdoe/removeRoles"
```

List Role Assignments of a Role Assignee

List all role assignments of a role assignee (i.e. a user or a group):

```
GET http://$SERVER/api/admin/assignments/assignees/$identifier
```

Note that `identifier` can contain slashes (e.g. `&ip/localhost-users`).

List Permissions a User Has on a Dataverse Collection or Dataset

List permissions a user (based on API Token used) has on a Dataverse collection or dataset:

```
GET http://$SERVER/api/admin/permissions/$identifier
```

The `$identifier` can be a Dataverse collection alias or database id or a dataset persistent ID or database id.

> **Note**
>
> Datasets can be selected using persistent identifiers. This is done by passing the constant `:persistentId` where the numeric id of the dataset is expected, and then passing the actual persistent id as a query parameter with the name `persistentId`.

Example: List permissions a user (based on API Token used) has on a dataset whose DOI is *10.5072/FK2/J8SJZB*:

```
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/J8SJZB

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/admin/permissions/:persistentId?
↪persistentId=$PERSISTENT_IDENTIFIER"
```

List Dataverse collections a user can act on based on their permissions

List Dataverse collections a user can act on based on a particular permission

```
GET http://$SERVER/api/users/$identifier/allowedCollections/$permission
```

> **Note**
>
> This API can only be called by an Administrator or by a User requesting their own list of accessible collections.

The `$identifier` is the username of the requested user. The `$permission` is the permission (tied to the roles) that gives the user access to the collection. Passing `$permission` as 'any' will return the collection as long as the user has any access/permission on the collection

```
export SERVER_URL=https://demo.dataverse.org
export $USERNAME=jsmith
export PERMISSION=PublishDataverse

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/users/$USERNAME/allowedCollections/
↪$PERMISSION"
```

Show Role Assignee

List a role assignee (i.e. a user or a group):

```
GET http://$SERVER/api/admin/assignee/$identifier
```

The `$identifier` should start with an `@` if it's a user. Groups start with `&`. "Built in" users and groups start with `:`. Private URL users start with `#`.

Saved Search

The Saved Search, Linked Dataverses, and Linked Datasets features are only accessible to superusers except for linking a dataset. The following API endpoints were added to help people with access to the "admin" API make use of these features in their current form. Keep in mind that they are partially experimental. The update of all saved search is run by a timer once a week (See *Saved Searches Links Timer*) so if you just created a saved search, you can run manually the `makelinks` endpoint that will find new dataverses and datasets that match the saved search and then link the search results to the dataverse in which the saved search is defined.

List All Saved Searches

```
export SERVER_URL=https://demo.dataverse.org

curl "$SERVER_URL/api/admin/savedsearches/list"
```

List a Saved Search by Database ID

```
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl "$SERVER_URL/api/admin/savedsearches/$ID"
```

Delete a Saved Search by Database ID

The `unlink=true` query parameter unlinks all links (linked dataset or Dataverse collection) associated with the deleted saved search. Use of this parameter should be well considered as you cannot know if the links were created manually or by the saved search. After deleting a saved search with `unlink=true`, we recommend running `/makelinks/all` just in case there was a dataset that was linked by another saved search. (Saved searches can link the same dataset.) Reindexing might be necessary as well.

```
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -X DELETE "$SERVER_URL/api/admin/savedsearches/$ID?unlink=true"
```

Execute a Saved Search and Make Links

Execute a saved search by database id and make links to Dataverse collections and datasets that are found. The JSON response indicates which Dataverse collections and datasets were newly linked versus already linked. The `debug=true` query parameter adds to the JSON response extra information about the saved search being executed (which you could also get by listing the saved search).

```
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -X PUT "$SERVER_URL/api/admin/savedsearches/makelinks/$ID?debug=true"
```

Execute All Saved Searches and Make Links

Execute all saved searches and make links to Dataverse collections and datasets that are found. `debug` works as described above. This happens automatically with a timer. For details, see *Saved Searches Links Timer* in the Admin Guide.

```
export SERVER_URL=https://demo.dataverse.org
export ID=1

curl -X PUT "$SERVER_URL/api/admin/savedsearches/makelinks/all?debug=true"
```

Dataset Integrity

Recalculate the UNF value of a dataset version, if it's missing, by supplying the dataset version database id:

```
POST http://$SERVER/api/admin/datasets/integrity/{datasetVersionId}/fixmissingunf
```

Datafile Integrity

Recalculate the check sum value value of a datafile, by supplying the file's database id and an algorithm (Valid values for $ALGORITHM include MD5, SHA-1, SHA-256, and SHA-512):

```
curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/admin/
↪computeDataFileHashValue/{fileId}/algorithm/$ALGORITHM"
```

Validate an existing check sum value against one newly calculated from the saved file:

```
curl -H "X-Dataverse-key:$API_TOKEN" -X POST "$SERVER_URL/api/admin/
↪validateDataFileHashValue/{fileId}"
```

Physical Files Validation in a Dataset

The following validates all the physical files in the dataset specified, by recalculating the checksums and comparing them against the values saved in the database:

```
$SERVER_URL/api/admin/validate/dataset/files/{datasetId}
```

It will report the specific files that have failed the validation. For example:

```
curl "http://localhost:8080/api/admin/validate/dataset/files/:persistentId/?
↪persistentId=doi:10.5072/FK2/XXXXX"
  {"dataFiles": [
                {"datafileId":2658,"storageIdentifier":"file://123-aaa","status":"valid"},
                {"datafileId":2659,"storageIdentifier":"file://123-bbb","status":"invalid
↪","errorMessage":"Checksum mismatch for datafile id 2669"},
                {"datafileId":2659,"storageIdentifier":"file://123-ccc","status":"valid"}
                ]
   }
```

These are only available to super users.

Update Checksums To Use New Algorithm

The fixity algorithm used on existing files can be changed by a superuser using this API call. An optional query parameter (num) can be used to limit the number of updates attempted (i.e. to do processing in batches). The API call will only update the algorithm and checksum for a file if the existing checksum can be validated against the file. Statistics concerning the updates are returned in the response to the API call with details in the log. The primary use for this API call is to update existing files after the algorithm used when uploading new files is changes - see - *:FileFixityChecksumAlgorithm*. Allowed values are MD5, SHA-1, SHA-256, and SHA-512

```
export ALG=SHA-256
export BATCHSIZE=1

curl "http://localhost:8080/api/admin/updateHashValues/$ALG"
curl "http://localhost:8080/api/admin/updateHashValues/$ALG?num=$BATCHSIZE"
```

Dataset Validation

Validate the dataset and its components (DatasetVersion, FileMetadatas, etc.) for constraint violations:

```
curl "$SERVER_URL/api/admin/validate/dataset/{datasetId}"
```

if validation fails, will report the specific database entity and the offending value. For example:

```
{"status":"OK","data":{"entityClassDatabaseTableRowId":"[DatasetVersion id:73]","field":
↪"archiveNote","invalidValue":"random text, not a url"}}
```

If the optional argument `variables=true` is specified, the API will also validate the metadata associated with any tabular data files found in the dataset specified. (For example: an invalid or empty variable name).

Validate all the datasets in the Dataverse installation, report any constraint violations found:

```
curl "$SERVER_URL/api/admin/validate/datasets"
```

If the optional argument `variables=true` is specified, the API will also validate the metadata associated with any tabular data files. (For example: an invalid or empty variable name). Note that validating all the tabular metadata may significantly increase the run time of the full validation pass.

This API streams its output in real time, i.e. it will start producing the output immediately and will be reporting on the progress as it validates one dataset at a time. For example:

```
{"datasets": [
              {"datasetId":27,"status":"valid"},
              {"datasetId":29,"status":"valid"},
              {"datasetId":31,"status":"valid"},
              {"datasetId":33,"status":"valid"},
              {"datasetId":35,"status":"valid"},
              {"datasetId":41,"status":"invalid","entityClassDatabaseTableRowId":
↪"[DatasetVersion id:73]","field":"archiveNote","invalidValue":"random text, not a url"}
↪,
              {"datasetId":57,"status":"valid"}
              ]
 }
```

Note that if you are attempting to validate a very large number of datasets in your Dataverse installation, this API may time out - subject to the timeout limit set in your app server configuration. If this is a production Dataverse installation serving large amounts of data, you most likely have that timeout set to some high value already. But if you need to increase it, it can be done with the asadmin command. For example:

```
asadmin set server-config.network-config.protocols.protocol.http-listener-1.http.request-
↪timeout-seconds=3600
```

Datafile Audit

Produce an audit report of missing files and FileMetadata for Datasets. Scans the Datasets in the database and verifies that the stored files exist. If the files are missing or if the FileMetadata is missing, this information is returned in a JSON response. The call will return a status code of 200 if the report was generated successfully. Issues found will be documented in the report and will not return a failure status code unless the report could not be generated:

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/admin/datafiles/auditFiles"
```

Optional Parameters are available for filtering the Datasets scanned.

For auditing the Datasets in a paged manner (firstId and lastId):

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/admin/datafiles/auditFiles?
↪firstId=0&lastId=1000"
```

Auditing specific Datasets (comma separated list):

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/admin/datafiles/auditFiles?
↪datasetIdentifierList=doi:10.5072/FK2/JXYBJS,doi:10.7910/DVN/MPU019"
```

Sample JSON Audit Response:

```
{
  "status": "OK",
  "data": {
     "firstId": 0,
     "lastId": 100,
     "datasetIdentifierList": [
         "doi:10.5072/FK2/XXXXXX",
         "doi:10.5072/FK2/JXYBJS",
         "doi:10.7910/DVN/MPU019"
     ],
     "datasetsChecked": 100,
     "datasets": [
          {
             "id": 6,
             "pid": "doi:10.5072/FK2/JXYBJS",
             "persistentURL": "https://doi.org/10.5072/FK2/JXYBJS",
             "missingFileMetadata": [
               {
                  "storageIdentifier": "local://1930cce4f2d-855ccc51fcbb",
                  "dataFileId": "7"
               }
             ]
         },
         {
             "id": 47731,
             "pid": "doi:10.5072/FK2/MPU019",
             "persistentURL": "https://doi.org/10.7910/DVN/MPU019",
             "missingFiles": [
               {
                  "storageIdentifier": "s3://dvn-cloud:298910",
                  "directoryLabel": "trees",
```

(continues on next page)

(continued from previous page)

```
                "label": "trees.png"
              }
            ]
          }
    ],
    "failures": [
        {
          "datasetIdentifier": "doi:10.5072/FK2/XXXXXX",
          "reason": "Not Found"
        }
    ]
  }
}
```

Workflows

List all available workflows in the system:

```
GET http://$SERVER/api/admin/workflows
```

Get details of a workflow with a given id:

```
GET http://$SERVER/api/admin/workflows/$id
```

Add a new workflow. Request body specifies the workflow properties and steps in JSON format. Specifically, the body of the message should be a JSON Object with a String "name" for the workflow and a "steps" JSON Array containing a JSON Object per workflow step. (See *Workflows* for the exiting steps and their required JSON representations.) Sample `json` files are available at `scripts/api/data/workflows/`:

```
POST http://$SERVER/api/admin/workflows
```

Delete a workflow with a specific id:

```
DELETE http://$SERVER/api/admin/workflows/$id
```

Warning

If the workflow designated by `$id` is a default workflow, a 403 FORBIDDEN response will be returned, and the deletion will be canceled.

List the default workflow for each trigger type:

```
GET http://$SERVER/api/admin/workflows/default/
```

Set the default workflow for a given trigger. This workflow is run when a dataset is published. The body of the PUT request is the id of the workflow. Trigger types are `PrePublishDataset, PostPublishDataset`:

```
PUT http://$SERVER/api/admin/workflows/default/$triggerType
```

Get the default workflow for `triggerType`. Returns a JSON representation of the workflow, if present, or 404 NOT FOUND.

```
GET http://$SERVER/api/admin/workflows/default/$triggerType
```

Unset the default workflow for `triggerType`. After this call, dataset releases are done with no workflow.

```
DELETE http://$SERVER/api/admin/workflows/default/$triggerType
```

Set the whitelist of IP addresses separated by a semicolon (`;`) allowed to resume workflows. Request body is a list of IP addresses allowed to send "resume workflow" messages to this Dataverse installation:

```
PUT http://$SERVER/api/admin/workflows/ip-whitelist
```

Get the whitelist of IP addresses allowed to resume workflows:

```
GET http://$SERVER/api/admin/workflows/ip-whitelist
```

Restore the whitelist of IP addresses allowed to resume workflows to default (localhost only):

```
DELETE http://$SERVER/api/admin/workflows/ip-whitelist
```

Metrics

Clear all cached metric results:

```
DELETE http://$SERVER/api/admin/clearMetricsCache
```

Clear a specific metric cache. Currently this must match the name of the row in the table, which is named *metricName*_*metricYYYYMM* (or just *metricName* if there is no date range for the metric). For example dataversesToMonth_2018-05:

```
DELETE http://$SERVER/api/admin/clearMetricsCache/$metricDbName
```

Inherit Dataverse Collection Role Assignments

Recursively applies the role assignments of the specified Dataverse collection, for the roles specified by the `:InheritParentRoleAssignments` setting, to all Dataverse collections contained within it:

```
GET http://$SERVER/api/admin/dataverse/{dataverse alias}/addRoleAssignmentsToChildren
```

Note: setting `:InheritParentRoleAssignments` will automatically trigger inheritance of the parent Dataverse collection's role assignments for a newly created Dataverse collection. Hence this API call is intended as a way to update existing child Dataverse collections or to update children after a change in role assignments has been made on a parent Dataverse collection.

Manage Available Standard License Terms

For more context about configuring licenses, see *Configuring Licenses* in the Installation Guide.

View the list of standard license terms that can be selected for a dataset:

```
export SERVER_URL=https://demo.dataverse.org
curl "$SERVER_URL/api/licenses"
```

View the details of the standard license with the database ID specified in `$ID`:

```
export ID=1
curl "$SERVER_URL/api/licenses/$ID"
```

Superusers can add a new license by posting a JSON file adapted from this example `add-license.json`. The `name` and `uri` of the new license must be unique. Sort order field is mandatory. If you are interested in adding a Creative Commons license, you are encouarged to use the JSON files under *Adding Creative Common Licenses*:

Licenses must have a "name" and "uri" and may have the following optional fields: "shortDescription", "iconUri", "rightsIdentifier", "rightsIdentifierScheme", "schemeUri", "languageCode", "active", "sortOrder". The "name" and "uri" are used to display the license in the user interface, with "shortDescription" and "iconUri" being used to enhance the display if available. The "rightsIdentifier", "rightsIdentifierScheme", and "schemeUri" should be added if the license is available from https://spdx.org . "languageCode" should be sent if the language is not in English ("en"). "active" is a boolean indicating whether the license should be shown to users as an option. "sortOrder" is a numeric value - licenses are shown in the relative numeric order of this value.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
curl -X POST -H 'Content-Type: application/json' -H "X-Dataverse-key:$API_TOKEN" --
↪upload-file add-license.json "$SERVER_URL/api/licenses"
```

Superusers can change whether an existing license is active (usable for new dataset versions) or inactive (only allowed on already-published versions) specified by the license `$ID`:

```
export STATE=true
curl -X PUT -H 'Content-Type: application/json' -H "X-Dataverse-key:$API_TOKEN" "$SERVER_
↪URL/api/licenses/$ID/:active/$STATE"
```

Superusers may change the default license by specifying the license `$ID`:

```
curl -X PUT -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/licenses/default/$ID"
```

Superusers can delete a license, provided it is not in use, by the license `$ID`:

```
curl -X DELETE -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/licenses/$ID"
```

Superusers can change the sorting order of a license specified by the license `$ID`:

```
export SORT_ORDER=100
curl -X PUT -H 'Content-Type: application/json' -H "X-Dataverse-key:$API_TOKEN" "$SERVER_
↪URL/api/licenses/$ID/:sortOrder/$SORT_ORDER"
```

List Dataset Templates

List all templates in the system.

```
GET http://$SERVER/api/admin/templates
```

List templates in a given dataverse by the dataverse's alias or id.

```
GET http://$SERVER/api/admin/templates/{alias or id}
```

Delete Dataset Template

A curl example using an `ID`

```
export SERVER_URL=https://demo.dataverse.org
export ID=24

curl -X DELETE "$SERVER_URL/api/admin/template/$ID"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -X DELETE "https://demo.dataverse.org/api/admin/template/24"
```

Request Signed URL

Dataverse has the ability to create signed URLs for it's API calls. A signature, which is valid only for the specific API call and only for a specified duration, allows the call to proceed with the authentication of the specified user. It is intended as an alternative to the use of an API key (which is valid for a long time period and can be used with any API call). Signed URLs were developed to support External Tools but may be useful in other scenarios where Dataverse or a third-party tool needs to delegate limited access to another user or tool. This API call allows a Dataverse superUser to generate a signed URL for such scenarios. The JSON input parameter required is an object with the following keys:

- `url` - the exact URL to sign, including api version number and all query parameters. Provide it in its URL-decoded form (for example `persistentId=doi:10.5072/FK2/ABC`): the signature is computed over the URL exactly as provided, and a URL signed in decoded form validates both when used as received and when parts of it get percent-encoded in flight. The URL must not already contain any of the reserved query parameters `until`, `user`, `method`, `token` (these are appended by the signing), `key` or `signed`; such a URL will be rejected with a 400 (Bad Request) response.
- `timeOut` - how long in minutes the signature should be valid for, default is 10 minutes
- `httpMethod` - which HTTP method is required, default is GET
- `user` - the user identifier for the account associated with this signature; if omitted, the URL is signed for the superuser making the call. A `user` that does not exist will be rejected with a 400 (Bad Request). The API call will succeed/fail based on whether the specified user has the required permissions.

A curl example using allowing access to a dataset's metadata

```
export SERVER_URL=https://demo.dataverse.org
export API_KEY=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export JSON='{"url":"https://demo.dataverse.org/api/v1/datasets/:persistentId/?
↪persistentId=doi:10.5072/FK2/J8SJZB","timeOut":5,"user":"alberteinstein"}'
```

(continues on next page)

(continued from previous page)

```
curl -H "X-Dataverse-key:$API_KEY" -H 'Content-Type:application/json' -d "$JSON" "
↪$SERVER_URL/api/admin/requestSignedUrl"
```

Use the returned `signedUrl` as received, treating it as an opaque string: do not parse and rebuild it or add query parameters, as reordering or inserting anything invalidates the signature. If you need a percent-encoded form (for example a search term containing spaces), append the returned signature parameters to the URL exactly as you originally constructed it.

The key used to sign the URL is created from the user's API token plus a server-side signing secret that is generated automatically at startup; see *dataverse.api.signing-secret* for details, including how to configure a persistent secret (required for signed URLs to survive server restarts and for multi-server installations).

Send Feedback To Contact(s) Admin API

This API call allows sending an email to the contacts for a collection, dataset, or datafile or to the support email address when no object is specified. The call is protected by the normal /admin API protections (limited to localhost or requiring a separate key), but does not otherwise limit the sending of emails. Administrators should be sure only trusted applications have access to avoid the potential for spam.

The call is a POST with a JSON object as input with four keys: - "targetId" - the id of the collection, dataset, or datafile. Persistent ids and collection aliases are not supported. (Optional) - "subject" - the email subject line - "body" - the email body to send - "fromEmail" - the email to list in the reply-to field. (Dataverse always sends mail from the system email, but does it "on behalf of" and with a reply-to for the specified user.)

A curl example using an `ID`

```
export SERVER_URL=http://localhost
export JSON='{"targetId":24, "subject":"Data Question", "body":"Please help me␣
↪understand your data. Thank you!", "fromEmail":"dataverseSupport@mailinator.com"}'

curl -X POST -H 'Content-Type:application/json' -d "$JSON" "$SERVER_URL/api/admin/
↪feedback"
```

Note that this call could be useful in coordinating with dataset authors (assuming they are also contacts) as an alternative/addition to the functionality provided by *Return a Dataset to Author*.

Send Feedback To Contact(s)

This API call allows sending an email to the contacts for a collection, dataset, or datafile or to the support email address when no object is specified. The call is protected from embedded html in the body as well as the ability to configure body size limits and rate limiting to avoid the potential for spam.

The call is a POST with a JSON object as input with four keys: - "targetId" - the id of the collection, dataset, or datafile. Persistent ids and collection aliases are not supported. (Optional) - "identifier" - the alias of a collection or the persistence id of a dataset or datafile. (Optional) - "subject" - the email subject line. (Required) - "body" - the email body to send (Required) - "fromEmail" - the email to list in the reply-to field. (Dataverse always sends mail from the system email, but does it "on behalf of" and with a reply-to for the specified user. Authenticated users will have the 'fromEmail' filled in from their profile if this field is not specified)

A curl example using an `ID`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
```

(continues on next page)

(continued from previous page)

```
export JSON='{"targetId":24, "subject":"Data Question", "body":"Please help me␣
↪understand your data. Thank you!"}'

curl -X POST -H "X-Dataverse-key:$API_KEY" -H 'Content-Type:application/json' -d "$JSON"
↪"$SERVER_URL/api/sendfeedback"
```

A curl example using a `Dataverse Alias or Dataset/DataFile PersistentId`

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export JSON='{"identifier":"root", "subject":"Data Question", "body":"Please help me␣
↪understand your data. Thank you!"}'

curl -X POST -H "X-Dataverse-key:$API_KEY" -H 'Content-Type:application/json' -d "$JSON"
↪"$SERVER_URL/api/sendfeedback"
```

Note that this call could be useful in coordinating with dataset authors (assuming they are also contacts) as an alternative/addition to the functionality provided by *Return a Dataset to Author*.

Reset Thumbnail Failure Flags

If Dataverse attempts to create a thumbnail image for an image or PDF file and the attempt fails, Dataverse will set a flag for the file to avoid repeated attempts to generate the thumbnail. For cases where the problem may have been temporary (or fixed in a later Dataverse release), the API calls below can be used to reset this flag for all files or for a given file.

```
export SERVER_URL=https://demo.dataverse.org
export FILE_ID=1234

curl -X DELETE $SERVER_URL/api/admin/clearThumbnailFailureFlag

curl -X DELETE $SERVER_URL/api/admin/clearThumbnailFailureFlag/$FILE_ID
```

Download File from /tmp

As a superuser:

```
GET /api/admin/downloadTmpFile?fullyQualifiedPathToFile=/tmp/foo.txt
```

Note that this API is probably only useful for testing.

5.6.18 MyData

The MyData API is used to get a list of just the datasets, collections (dataverses), or datafiles an authenticated user has a role on.

The API excludes collections linked to an harvesting client. This results in a known issue where regular datasets in harvesting collections are missing from the results.

Here is a curl example.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export ROLE_ID1=6
export ROLE_ID2=8
export DVTYPE1=Dataset
export DVTYPE2=Dataverse
export PUBLISHED_STATE1=Unpublished
export PUBLISHED_STATE2=Published
export PER_PAGE=10

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/mydata/retrieve?role_ids=$ROLE_ID1&
↪role_ids=$ROLE_ID2&dvobject_types=$DVTYPE1&dvobject_types=$DVTYPE2&published_states=
↪$PUBLISHED_STATE1&published_states=$PUBLISHED_STATE2&per_page=$PER_PAGE"
```

The fully expanded example above (without environment variables) looks like this:

```
curl -H "X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" "https://demo.dataverse.
↪org/api/mydata/retrieve?role_ids=6&role_ids=8&dvobject_types=Dataset&published_
↪states=Unpublished&published_states=Published&per_page=10"
```

Parameters:

- **`role_ids`: Roles are customizable. Multiple "role_ids" parameters can be used to include several roles. Standard roles include:**
  - `1` = Admin
  - `2` = File Downloader
  - `3` = Dataverse + Dataset Creator
  - `4` = Dataverse Creator
  - `5` = Dataset Creator
  - `6` = Contributor
  - `7` = Curator
  - `8` = Member
- **`dvobject_types`: Type of object. Multiple "dvobject_types" parameters can be used to include several types. Possible values:**
  - `Dataverse`
  - `Dataset`
  - `DataFile`
- **`published_states`: State of the object. Multiple "published_states" parameters can be used to include several states. Possible values:**
  - `Published`
  - `Unpublished`
  - `Draft`
  - `Deaccessioned`
  - `In+Review` (the + represents a space)
- `mydata_search_term`: A string used to search for specific data within the user's MyData collection.

- `selected_page`: The page number of results to return (used for pagination).
- `per_page`: Number of results returned per page.
- `order`: The order in which to sort. Can either be "asc" or "desc".
- `fq`: A filter query (Solr syntax) to narrow the list returned. Multiple "fq" parameters can be used.

MyData Collection List

The MyData Collection List API is used to get a list of the collections an authenticated user can create a Dataset in. Param userIdentifier={userName} is used by a superuser to get the collections for a specific user.

A curl example listing collections:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/mydata/retrieve/collectionList"
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/mydata/retrieve/collectionList?
↪userIdentifier=anotherUser"
```

5.7 Metrics API

The Metrics API provides counts of downloads, datasets created, files uploaded, user accounts created, and more, as described below. The Dataverse Software also includes aggregate counts of Make Data Count metrics (described in the *Make Data Count* section of the Admin Guide and available per-Dataset through the *Native API*). A table of all the endpoints is listed below.

Contents:

- *Categories*
- *Return Formats*
- *Filtering with Query Parameters*
  - *parentAlias*
  - *dataLocation*
  - *country*
- *Endpoint Table*
- *Related API Endpoints*

Note

The Metrics API can be used from scripts running in web browsers, as it allows cross-origin resource sharing (CORS).

Note

For all metrics *besides* Past Days Count (`/pastDays/$days`) - recalculated daily, and (`/toMonth/$month`) for prior months - never recalculated, the setting `MetricsCacheTimeoutMinutes` defines how long the cached value will be returned by subsequent queries.

5.7.1 Categories

The Metrics API includes several categories of endpoints that provide different ways of understanding the evolution of metrics over time:

- Total - an aggregate count over all-time:
  - Form: GET https://\protect\T1\textdollarSERVER/api/info/metrics/\protect\T1\textdollartype
  - where `$type` can be set, for example, to `dataverses` (Dataverse collections), `datasets`, `files`, `downloads` or `accounts`.
  - Example: `curl https://demo.dataverse.org/api/info/metrics/downloads`
  - Return: Most of these calls return a simple JSON object with a `count` whose value is the metric's total count. Some calls, such as `filedownloads` return aggregate metrics per item (e.g. per file or by subject) as a JSONArray or CSV (see Return Formats below)
- To-Month - a count of various objects in dataverse up to and including a specified month `$YYYY-DD` in YYYY-MM format (e.g. `2018-01`):
  - Form: GET https://\protect\T1\textdollarSERVER/api/info/metrics/\protect\T1\textdollartype/toMonth/\protect\T1\textdollarYYYY-DD
  - where `$type` can be set, for example, to `dataverses` (Dataverse collections), `datasets`, `files`, `downloads` or `accounts`.
  - Example: `curl https://demo.dataverse.org/api/info/metrics/dataverses/toMonth/2018-01`
  - Return: Most of these calls return a simple JSON object with a `count` whose value is the metric's total count. One variant, `/api/info/metrics/datasets/bySubject/toMonth` return aggregate metrics per Dataset Subject as a JSONArray or CSV (see Return Formats below)
- Past Days - a count of various objects in a Dataverse installation for the past `$days` (e.g. `30`):
  - Form: GET https://\protect\T1\textdollarSERVER/api/info/metrics/\protect\T1\textdollartype/pastDays/\protect\T1\textdollardays
  - where `$type` can be set, for example, to `dataverses` (Dataverse collections), `datasets`, `files`, `downloads` or `accounts`.
  - Example: `curl https://demo.dataverse.org/api/info/metrics/datasets/pastDays/30`
  - Return: A simple JSON object with a `count` whose value is the metric's total count.
- Monthly - a time series of the metric with aggregate values per month with counts up to and including the given month:
  - Form: GET https://\protect\T1\textdollarSERVER/api/info/metrics/\protect\T1\textdollartype/monthly
  - where `$type` can be set, for example, to `dataverses` (Dataverse collections), `datasets`, `files`, `downloads` or `accounts`.
  - Example: `curl https://demo.dataverse.org/api/info/metrics/downloads/monthly`

  - Return: A JSON Array or CSV file with an array of counts per month. Variants of this this category that provide a time series per object return that information in the same formats (JSON array, CSV) with one time-series column per item (see Return Formats below).

- Tree - endpoints that describe the structure of the tree of published Dataverses, as of now or as of the end of a specified month. The monthly version could be used to show growth of the Dataverse instance over time, but the initial use case for these endpoints was to provide a means to list the tree in a selection widget to scope the metrics displayed in the local Dataverse metrics page in the dataverse-metrics app.
  - Form: GET https://\protect\T1\textdollarSERVER/api/info/metrics/tree
  - or
  - Form: GET https://\protect\T1\textdollarSERVER/api/info/metrics/tree/toMonth/\protect\T1\textdollarYYYY-DD
  - Example: `curl https://demo.dataverse.org/api/info/metrics/tree`
  - Return: A nested JSON array containing JSON objects for each Dataverse collection with key/values for id, ownerId, alias, depth, and name, and a JSON array containing analogous objects for Dataverse collections within the current one.

5.7.2 Return Formats

There are a number of API calls that provide time series, information reported per item (e.g. per dataset, per file, by subject, by category, and by file Mimetype), or both (time series per item). Because these calls all report more than a single number, the API provides two optional formats for the return that can be selected by specifying an HTTP Accept Header for the desired format:

- application/json - a JSON array of objects. For time-series, the objects include key/values for the `date` and `count` for that month. For per-item calls, the objects include the item (e.g. for a subject), or it's id/pid (for a dataset or datafile (which may/may not not have a PID)). For timeseries per-item, the objects also include a date. In all cases, the response is a single array.
  - Example: `curl -H 'Accept:application/json' https://demo.dataverse.org/api/info/metrics/downloads/monthly`
- comma-separated-value (CSV) - a CSV file with rows corresponding to each JSON object in the application/json format. Column headers are included (e.g. `date,count` or `subject,count` or `date,pid,id,count` (for a time series per file)).
  - Example: `curl -H 'Accept:text/csv' https://demo.dataverse.org/api/info/metrics/downloads/monthly`
  - The default format is CSV, so `curl https://demo.dataverse.org/api/info/metrics/downloads/monthly`, or typing this URL into a browser return the CSV format.

5.7.3 Filtering with Query Parameters

To further tailor your metric, query parameters can be provided. On relevant endpoints, these query parameters can be used together.

parentAlias

Specifies which Dataverse sub-collection the metric should be collected for. Not including this parameter gathers metrics for the entire instance.

Example: `curl https://demo.dataverse.org/api/info/metrics/datasets/?parentAlias=abc` would return the number of datasets in the Dataverse collection with alias 'abc' and in sub-collections within it.

dataLocation

Specifies whether the metric should query `local` data, `remote` data (e.g. harvested), or `all` data when getting results. Only works for dataset metrics.

Example: `curl https://demo.dataverse.org/api/info/metrics/datasets/?dataLocation=remote`

country

The Make Data Count endpoints are also able to filter results by Country (specified using the ISO 3166 Country codes)

Example: `curl https://demo.dataverse.org/api/info/metrics/makeDataCount/viewsTotal?country=au`

5.7.4 Endpoint Table

The following table lists the available metrics endpoints (not including the Make Data Counts endpoints for a single dataset which are part of the *Native API*) along with additional notes about them.

Table 1: Metrics Endpoints

endpoint	variable	format	scope	limits	ca	meaning	notes
/api/info/metrics/dataverses	coun	jso	collection subtree	published	y	as of now/total	collection subtree means you can get info for the instance or with ?parentAlias={alias} can optionally specify a dataverse which should be used to scope the query.
/api/info/metrics/dataverses/toMonth/{yy MM}	coun	jso	collection subtree	published	y	cumulative up to month specified	
/api/info/metrics/dataverses/monthly	date, coun	jso csv	collection subtree	published	y	monthly cumulative time-series from first date of first entry to now	

continues on next page

Table 1 – continued from previous page

endpoint	variables	formats	scope	limits	caching	meaning	notes
/api/info/metrics/dataverses/pastDays/{n}	count	json	collection subtree	published	y	aggregate count for past n days	
/api/info/metrics/dataverses/byCategory	category, count	json csv	collection subtree	published	y	total count per category	
/api/info/metrics/dataverses/bySubject	subject, count	json csv	collection subtree	all	y	total count per subject	
/api/info/metrics/datasets	count	json	collection subtree	released, choice of all, local or remote (harvested)	y	as of now/total	released means only currently released dataset versions (not unpublished or DEACCESSIONED versions)
/api/info/metrics/datasets/toMonth/{yyyy-MM}	count	json	collection subtree	released, choice of all, local or remote (harvested)	y	cumulative up to month specified	
/api/info/metrics/datasets/monthly	date, count	json csv	collection subtree	released, choice of all, local or remote (harvested)	y	monthly cumulative time-series from first date of first entry to now	released means only currently released dataset versions (not unpublished or DEACCESSIONED versions)

continues on next page

Table 1 – continued from previous page

endpoint	variables	formats	scope	limits	ca	meaning	notes
/api/info/metrics/datasets/pastDays/{n}	count	json	collection subtree	released, choice of all, local or remote (harvested)	y	aggregate count for past n days	
/api/info/metrics/datasets/bySubject	subject, count	json csv	collection subtree	released, choice of all, local or remote (harvested)	y	total count per subject	
/api/info/metrics/datasets/bySubject/toMo MM}	subject, count	json csv	collection subtree	released, choice of all, local or remote (harvested)	y	cumulative cont per subject up to month specified	
/api/info/metrics/files	count	json	collection subtree	in released dataset	y	as of now/total	
/api/info/metrics/files/toMonth/{yyyy-MM}	count	json	collection subtree	in released dataset	y	cumulative up to month specified	
/api/info/metrics/files/monthly	date, count	json csv	collection subtree	in released dataset	y	monthly cumulative time-series from first date of first entry to now	date is the month when the first version containing the file was released (or created for harvested versions)

continues on next page

Table 1 – continued from previous page

endpoint	variables	formats	scope	limits	ca	meaning	notes
/api/info/metrics/files/pastDays/{n}	count	json	collection subtree	in released datasets	y	aggregate count for past n days	
/api/info/metrics/files/byType	mimetype, count, size	json, csv	collection subtree	in released datasets	y	current totals	
/api/info/metrics/files/byType/monthly	date, mimetype, count, size	json, csv	collection subtree	in released datasets	y	monthly cumulative time-series from first date of first entry to now	data for a specific mimetype is only listed starting with the first month there are files of that type
/api/info/metrics/downloads	count	json	collection subtree	published	y	as of now/total	published for downloads means 'recorded in guestbookresponse' which occurs for any files that were ever in a published version, even if that version is now DEACCESSIONED, the file isn't in a current version, etc.
/api/info/metrics/downloads/toMonth/{yyyyMM}	count	json	collection subtree	published	y	cumulative up to month specified	downloads from versions that do not have a releasetime (from older Dataverse versions) are included in this cumulative count and the total as of now (line above)
/api/info/metrics/downloads/pastDays/{n}	count	json	collection subtree	published	y	aggregate count for past n days	
/api/info/metrics/downloads/monthly	date, count	json, csv	collection subtree	published	y	monthly cumulative time-series from first date of first entry to now	counts from dataset versions with no releasetime (legacy from old Dataverse versions) are counted as occuring in the month prior to the first count that does have a date

continues on next page

Table 1 – continued from previous page

endpoint	variables	formats	scope	limits	ca	meaning	notes
/api/info/metrics/filedownloads	count by id, pid	json csv	collection subtree	published	y	as of now/totals	download counts per file id. PIDs are also included in output if they exist
/api/info/metrics/filedownloads/toMonth/MM}	count by id, pid	json csv	collection subtree	published	y	cumulative up to month specified	download counts per file id to the specified month. PIDs are also included in output if they exist
/api/info/metrics/filedownloads/monthly	date, count id, pid	json csv	collection subtree	published	y	monthly cumulative time-series by file id, pid from first date of first entry to now	unique downloads per month by file (id, pid) sorted in decreasing order of counts
/api/info/metrics/makeDataCount/{metric	count	json	collection subtree, optionally also by {country}	published, MDC	y	count for specified {metric} as of now/total	published means in the mdc logs which are not created for unpublished datasets, so this is filtered like downloads and includes counts from DEACCESSED, old versions.
/api/info/metrics/makeDataCount/{metric MM}	count	json	collection subtree, optionally also by {country}	published, MDC	y	cumulative count for specified {metric} through specified month	These metrics are also limited by the MDC start date and by MDC filtering done by counter-processor

continues on next page

Table 1 – continued from previous page

endpoint	variables	formats	scope	limits	ca	meaning	notes
/api/info/metrics/makeDataCount/{metric	date, count	json, csv	collection subtree, optionally also by {country}	published, MDC	y	monthly cumulative time-series of counts for specified {metric}	These metrics are also limited by the MDC start date and by MDC filtering done by counter-processor
/api/info/metrics/uniquedownloads	pid, count	json	collection subtree	published	y	total count of unique users who have downloaded from the datasets in scope	The use case for this metric (uniquedownloads) is to more fairly assess which datasets are getting downloaded/used by only counting each users who downloads any file from a dataset as one count (versus downloads of multiple files or repeat downloads counting as multiple counts which adds a bias for large datasets and/or use patterns where a file is accessed repeatedly for new analyses)
/api/info/metrics/uniquedownloads/month	date, pid, count	json, csv	collection subtree	published	y	monthly cumulative time-series of unique user counts for datasets in the dataverse scope	

continues on next page

Table 1 – continued from previous page

endpoint	variables	formats	scope	limits	caches	meaning	notes
/api/info/metrics/uniquedownloads/toMonth/{yyyy-MM}	pid, count	json	collection subtree	published	y	cumulative count of unique users who have downloaded from the datasets in scope through specified month	
/api/info/metrics/uniquefiledownloads	count by id, pid	json csv	collection subtree	published	y	as of now/totals	unique download counts per file id. PIDs are also included in output if they exist
/api/info/metrics/uniquefiledownloads/monthly	date, count id, pid	json csv	collection subtree	published	y	monthly cumulative time-series by file id, pid from first date of first entry to now	unique downloads per month by file (id, pid) sorted in decreasing order of counts
/api/info/metrics/uniquefiledownloads/toMonth/{yyyy-MM}	count by id, pid	json csv	collection subtree	published	y	cumulative up to month specified	unique download counts per file id to the specified month. PIDs are also included in output if they exist

continues on next page

Table 1 – continued from previous page

endpoint	variables	format	scope	limits	cache	meaning	notes
/api/info/metrics/tree	id, owner alias, depth name children	json	collection subtree	published	y	tree of dataverses starting at the root or a specified parentAlias with their id, owner id, alias, name, a computed depth, and array of children dataverses	underlying code can also include draft dataverses, this is not currently accessible via api, depth starts at 0
/api/info/metrics/tree/toMonth/{yyyy-MM}	id, owner alias, depth name children	json	collection subtree	published	y	tree of dataverses in existence as of specified date starting at the root or a specified parentAlias with their id, owner id, alias, name, a computed depth, and array of children dataverses	underlying code can also include draft dataverses, this is not currently accessible via api, depth starts at 0
/api/info/metrics/accounts	count	json	Dataverse installation	all	y	as of now/totals	

continues on next page

Table 1 – continued from previous page

endpoint	variables	format	scope	limits	ca	meaning	notes
/api/info/metrics/accounts/toMonth/{yyyy-MM}	count	json	Dataverse installation	all	y	cumulative up to month specified	
/api/info/metrics/accounts/pastDays/{n}	count	json	Dataverse installation	all	y	aggregate count for past n days	
/api/info/metrics/accounts/monthly	date, count	json csv	Dataverse installation	all	y	monthly cumulative time-series from first date of first entry to now	

5.7.5 Related API Endpoints

The following endpoints are not under the metrics namespace but also return counts:

- *Getting File Download Count*

5.8 SWORD API

SWORD stands for "Simple Web-service Offering Repository Deposit" and is a "profile" of AtomPub (RFC 5023) which is a RESTful API that allows non-Dataverse Software to deposit files and metadata into a Dataverse installation. *Client libraries* are available in Python, Javascript, Java, R, Ruby, and PHP.

Contents:

- *About*
- *Authentication for SWORD*
- *Backward incompatible changes*
- *New features as of v1.1*
- *curl examples*
  - *Retrieve SWORD service document*
  - *Create a dataset with an Atom entry*

        * *Dublin Core Terms (DC Terms) Qualified Mapping - Dataverse Project DB Element Crosswalk*
    - *List datasets in a Dataverse Collection*
    - *Add files to a dataset with a zip file*
    - *Display a dataset atom entry*
    - *Display a dataset statement*
    - *Delete a file by database id*
    - *Replacing metadata for a dataset*
    - *Delete a dataset*
    - *Determine if a Dataverse Collection has been published*
    - *Publish a Dataverse Collection*
    - *Publish a dataset*
- *Known issues*
- *Bug fixes in v1.1*
- *Client libraries*

5.8.1 About

Introduced in Dataverse Network (DVN) 3.6, the SWORD API was formerly known as the "Data Deposit API" and `data-deposit/v1` appeared in the URLs. For backwards compatibility these URLs continue to work (with deprecation warnings). Due to architectural changes and security improvements (especially the introduction of API tokens) in Dataverse Software 4.0, a few backward incompatible changes were necessarily introduced and for this reason the version has been increased to `v1.1`. For details, see *Backward incompatible changes*.

The Dataverse Software implements most of SWORDv2, which is specified at https://swordapp.github.io/SWORDv2-Profile/SWORDProfile.html . Please reference the SWORDv2 specification for expected HTTP status codes (i.e. 201, 204, 404, etc.), headers (i.e. "Location"), etc.

As a profile of AtomPub, XML is used throughout SWORD. As of Dataverse Software 4.0 datasets can also be created via JSON using the "native" API. SWORD is limited to the dozen or so fields listed below in the crosswalk, but the native API allows you to populate all metadata fields available in a Dataverse installation.

5.8.2 Authentication for SWORD

The *API Tokens and Authentication* section describes how to pass API tokens for the "native" API (as a header or query parameter) but SWORD uses a different mechanism known as HTTP Basic Authentication (RFC 7617).

HTTP Basic Authentication commonly makes use of both a username and a password but the SWORD API uses only a username that is populated with the user's API token. The password should be blank or empty.

Clients such as `curl` expect both a username and a password separated by a colon. With `curl`, the way to indicate that the password should be blank or empty is to include the colon at the end of the username (the API token) like this:

```
curl -u 54b143b5-d001-4254-afc0-a1c0f6a5b5a7:
```

All the curl examples below take this form but instead of showing an API token like above, a Bash environment variable called `$API_TOKEN` is shown instead like this:

```
curl -u $API_TOKEN:
```

5.8.3 Backward incompatible changes

For better security than in DVN 3.x, usernames and passwords are no longer accepted. The use of an API token is required.

Differences in Dataverse Software 4 from DVN 3.x lead to a few minor backward incompatible changes in the Dataverse Software implementation of SWORD, which are listed below. Old `v1` URLs should continue to work but the `Service Document` will contain a deprecation warning and responses will contain `v1.1` URLs. See also *Known issues*.

- Newly required fields when creating/editing datasets for compliance with the Joint Declaration for Data Citation principles.
  - `dcterms:creator` (maps to authorName)
  - `dcterms:description`
- Deaccessioning is no longer supported. An alternative will be developed at https://github.com/IQSS/dataverse/issues/778
- The Service Document will show a single API Terms of Use rather than root level and Dataverse collection level Deposit Terms of Use.
- As of Dataverse Software 5.10, `NONE` is no longer supported as a license.

5.8.4 New features as of v1.1

- The Dataverse Software supports API tokens and requires them to be used for APIs instead of a username and password. In the `curl` examples below, you will see `curl -u $API_TOKEN:` showing that you should send your API token as the username and nothing as the password. For example, `curl -u 54b143b5-d001-4254-afc0-a1c0f6a5b5a7:`. See also *Authentication for SWORD*.
- SWORD operations no longer require "admin" permission. In order to use any SWORD operation in DVN 3.x, you had to be an "admin" on a Dataverse collection (the container for your dataset) and similar rules were applied in Dataverse Software 4.4 and earlier (the `EditDataverse` permission was required). The SWORD API has now been fully integrated with the Dataverse Software permission model such that any action you have permission to perform in the GUI or "native" API you are able to perform via SWORD. This means that even a user with a "Contributor" role can operate on datasets via SWORD. Note that users with the "Contributor" role do not have the `PublishDataset` permission and will not be able publish their datasets via any mechanism, GUI or API.
- Dataverse collections can be published via SWORD.
- Datasets versions will only be increased to the next minor version (i.e. 1.1) rather than a major version (2.0) if possible. This depends on the nature of the change. Adding or removing a file, for example, requires a major version bump.
- "Author Affiliation" can now be populated with an XML attribute. For example: `<dcterms:creator affiliation="Coffee Bean State University">Stumptown, Jane</dcterms:creator>`
- "Contributor" can now be populated and the "Type" (Editor, Funder, Researcher, etc.) can be specified with an XML attribute. For example: `<dcterms:contributor type="Funder">CaffeineForAll</dcterms:contributor>`
- "License" can now be set with `dcterms:license` and the possible values determined by the installation ("CC0 1.0" and "CC BY 4.0" by default). "License" interacts with "Terms of Use" (`dcterms:rights`) in that if you include `dcterms:rights` in the XML and don't include `dcterms:license`, the license will be "Custom Dataset Terms" and "Terms of Use" will be populated. If you don't include `dcterms:rights`, the default license will

be used. It is invalid to specify a license and also include `dcterms:rights`; an error will be returned. For backwards compatibility, `dcterms:rights` is allowed to be blank (i.e. `<dcterms:rights></dcterms:rights>`) but blank values will not be persisted to the database and the license will be set to "Custom Dataset Terms". Note that if admins of an installation have disabled "Custom Dataset Terms" you will get an error if you try to pass `dcterms:rights`.

- "Contact E-mail" is automatically populated from dataset owner's email.
- "Subject" uses our controlled vocabulary list of subjects. This list is in the Citation Metadata of our User Guide > Metadata References. Otherwise, if a term does not match our controlled vocabulary list, it will put any subject terms in "Keyword". If Subject is empty it is automatically populated with "N/A".
- Zero-length files are now allowed (but not necessarily encouraged).
- "Depositor" and "Deposit Date" are auto-populated.

5.8.5 curl examples

Retrieve SWORD service document

The service document enumerates the Dataverse collections (also "collections" from a SWORD perspective) the user can deposit data into. The "collectionPolicy" element for each Dataverse collections contains the Terms of Use. Any user with an API token can use this API endpoint. Institution-wide Shibboleth groups are not respected because membership in such a group can only be set via a browser.

```
curl -u $API_TOKEN: https://$HOSTNAME/dvn/api/data-deposit/v1.1/swordv2/service-document
```

Create a dataset with an Atom entry

To create a dataset, you must have the "Dataset Creator" role (the `AddDataset` permission) on a Dataverse collection. Practically speaking, you should first retrieve the service document to list the Dataverse collections into which you are authorized to deposit data.

```
curl -u $API_TOKEN: --data-binary "@path/to/atom-entry-study.xml" -H "Content-Type:␣
↪application/atom+xml" https://$HOSTNAME/dvn/api/data-deposit/v1.1/swordv2/collection/
↪dataverse/$DATAVERSE_ALIAS
```

Example Atom entry (XML)

```
<?xml version="1.0"?>
<entry xmlns="http://www.w3.org/2005/Atom"
       xmlns:dcterms="http://purl.org/dc/terms/">
   <!-- some embedded metadata -->
   <dcterms:title>Roasting at Home</dcterms:title>
   <dcterms:creator>Peets, John</dcterms:creator>
   <dcterms:creator affiliation="Coffee Bean State University">Stumptown, Jane</
↪dcterms:creator>
   <!-- Dataverse controlled vocabulary subject term -->
   <dcterms:subject>Chemistry</dcterms:subject>
   <!-- keywords -->
   <dcterms:subject>coffee</dcterms:subject>
   <dcterms:subject>beverage</dcterms:subject>
   <dcterms:subject>caffeine</dcterms:subject>
   <dcterms:description>Considerations before you start roasting your own coffee at home.
```

(continues on next page)

(continued from previous page)

```
↪</dcterms:description>
   <!-- Producer with financial or admin responsibility of the data -->
   <dcterms:publisher>Coffee Bean State University</dcterms:publisher>
   <dcterms:contributor type="Funder">CaffeineForAll</dcterms:contributor>
   <!-- production date -->
   <dcterms:date>2013-07-11</dcterms:date>
   <!-- kind of data -->
   <dcterms:type>aggregate data</dcterms:type>
   <!-- List of sources of the data collection-->
   <dcterms:source>Stumptown, Jane. 2011. Home Roasting. Coffeemill Press.</
↪dcterms:source>
   <!-- related materials -->
   <dcterms:relation>Peets, John. 2010. Roasting Coffee at the Coffee Shop. Coffeemill␣
↪Press</dcterms:relation>
   <!-- geographic coverage -->
   <dcterms:coverage>United States</dcterms:coverage>
   <dcterms:coverage>Canada</dcterms:coverage>
   <!-- license and restrictions -->
   <dcterms:license>CC0 1.0</dcterms:license>
   <!--<dcterms:rights>Downloader will not use the Materials in any way prohibited by␣
↪applicable laws.</dcterms:rights>-->
   <!-- related publications -->
   <dcterms:isReferencedBy holdingsURI="http://dx.doi.org/10.1038/dvn333" agency="DOI"␣
↪IDNo="10.1038/dvn333">Peets, J., &amp;amp; Stumptown, J. (2013). Roasting at Home. New␣
↪England Journal of Coffee, 3(1), 22-34.</dcterms:isReferencedBy>
</entry>
```

Dublin Core Terms (DC Terms) Qualified Mapping - Dataverse Project DB Element Crosswalk

DC (terms: namespace)	Dataverse Software DB Element	Require	Note
dcterms:title	title	Y	Title of the Dataset.
dcterms:creato	authorName (LastName, FirstName)	Y	Author(s) for the Dataset.
dcterms:subjec	subject (Controlled Vocabulary) OR keyword	Y	Controlled Vocabulary list is in our User Guide > Metadata References.
dcterms:descri	dsDescriptionValue	Y	Describing the purpose, scope or nature of the Dataset. Can also use dcterms:abstract.
dcterms:publis	producerName		Person or agency financially or administratively responsible for the Dataset
dcterms:contri	datasetContactEmail	Y	Contact Email is required so will need to add an attribute type="Contact". Also used for Funder: add attribute type="Funder" which maps to contributorName.
dcterms:date	productionDate (YYYY-MM-DD or YYYY-MM or YYYY)		Production date of Dataset.
dcterms:type	kindOfData		Type of data included in the file: survey data, census/enumeration data, aggregate data, clinical.
dcterms:source	dataSources		List of books, articles, data files if any that served as the sources for the Dataset.
dcterms:relatio	relatedMaterial		Any related material (journal article citation is not included here - see: dcterms:isReferencedBy below).
dcterms:covera	otherGeographicCoverage		General information on the geographic coverage of the Dataset.
dcterms:licens	license		Set the license. Alternatively, use the dcterms:rights field instead.
dcterms:rights	termsofuse		If not using dcterms:license, enter any terms of use or restrictions for the Dataset.
dcterms:isRefe	publicationCitation		The publication (journal article, book, other work) that uses this dataset (include citation, permanent identifier (DOI), and permanent URL).

List datasets in a Dataverse Collection

You must have permission to add datasets in a Dataverse collection (the Dataverse collection should appear in the service document) to list the datasets inside. Institution-wide Shibboleth groups are not respected because membership in such a group can only be set via a browser.

```
curl -u $API_TOKEN: https://$HOSTNAME/dvn/api/data-deposit/v1.1/swordv2/collection/
↪dataverse/$DATAVERSE_ALIAS
```

Add files to a dataset with a zip file

You must have `EditDataset` permission (Contributor role or above such as Curator or Admin) on the dataset to add files.

```
curl -u $API_TOKEN: --data-binary @path/to/example.zip -H "Content-Disposition:␣
↪filename=example.zip" -H "Content-Type: application/zip" -H "Packaging: http://purl.
↪org/net/sword/package/SimpleZip" https://$HOSTNAME/dvn/api/data-deposit/v1.1/swordv2/
↪edit-media/study/doi:TEST/12345
```

Display a dataset atom entry

You must have `ViewUnpublishedDataset` permission (Contributor role or above such as Curator or Admin) on the dataset to view its Atom entry.

Contains data citation (bibliographicCitation), alternate URI (persistent URI of study), edit URI, edit media URI, statement URI.

```
curl -u $API_TOKEN: https://$HOSTNAME/dvn/api/data-deposit/v1.1/swordv2/edit/study/
↪doi:TEST/12345
```

Display a dataset statement

Contains title, author, feed of file entries, latestVersionState, locked boolean, updated timestamp. You must have `ViewUnpublishedDataset` permission (Contributor role or above such as Curator or Admin) on the dataset to display the statement.

```
curl -u $API_TOKEN: https://$HOSTNAME/dvn/api/data-deposit/v1.1/swordv2/statement/study/
↪doi:TEST/12345
```

Delete a file by database id

You must have `EditDataset` permission (Contributor role or above such as Curator or Admin) on the dataset to delete files.

```
curl -u $API_TOKEN: -X DELETE https://$HOSTNAME/dvn/api/data-deposit/v1.1/swordv2/edit-
↪media/file/123
```

Replacing metadata for a dataset

Please note that **ALL** metadata (title, author, etc.) will be replaced, including fields that can not be expressed with "dcterms" fields. You must have `EditDataset` permission (Contributor role or above such as Curator or Admin) on the dataset to replace metadata.

```
curl -u $API_TOKEN: --upload-file "path/to/atom-entry-study2.xml" -H "Content-Type:␣
↪application/atom+xml" https://$HOSTNAME/dvn/api/data-deposit/v1.1/swordv2/edit/study/
↪doi:TEST/12345``
```

Delete a dataset

You must have the `DeleteDatasetDraft` permission (Contributor role or above such as Curator or Admin) on the dataset to delete it. Please note that if the dataset has never been published you will be able to delete it completely but if the dataset has already been published you will only be able to delete post-publication drafts, never a published version.

```
curl -u $API_TOKEN: -i -X DELETE https://$HOSTNAME/dvn/api/data-deposit/v1.1/swordv2/
↪edit/study/doi:TEST/12345
```

Determine if a Dataverse Collection has been published

This API endpoint is the same as the "list datasets in a Dataverse collection" endpoint documented above and the same permissions apply but it is documented here separately to point out that you can look for a boolean called `dataverseHasBeenReleased` to know if a Dataverse collection has been released, which is required for publishing a dataset.

```
curl -u $API_TOKEN: https://$HOSTNAME/dvn/api/data-deposit/v1.1/swordv2/collection/
↪dataverse/$DATAVERSE_ALIAS
```

Publish a Dataverse Collection

The `cat /dev/null` and `--data-binary @-` arguments are used to send zero-length content to the API, which is required by the upstream library to process the `In-Progress: false` header. You must have the `PublishDataverse` permission (Admin role) on the Dataverse collection to publish it.

```
cat /dev/null | curl -u $API_TOKEN: -X POST -H "In-Progress: false" --data-binary @-␣
↪https://$HOSTNAME/dvn/api/data-deposit/v1.1/swordv2/edit/dataverse/$DATAVERSE_ALIAS
```

Publish a dataset

The `cat /dev/null` and `--data-binary @-` arguments are used to send zero-length content to the API, which is required by the upstream library to process the `In-Progress: false` header. You must have the `PublishDataset` permission (Curator or Admin role) on the dataset to publish it.

```
cat /dev/null | curl -u $API_TOKEN: -X POST -H "In-Progress: false" --data-binary @-␣
↪https://$HOSTNAME/dvn/api/data-deposit/v1.1/swordv2/edit/study/doi:TEST/12345
```

5.8.6 Known issues

- Deleting a file from a published version (not a draft) creates a draft but doesn't delete the file: https://github.com/IQSS/dataverse/issues/2464
- The Service Document does not honor groups within groups: https://github.com/IQSS/dataverse/issues/3056
- Should see all the fields filled in for a dataset regardless of what the parent Dataverse collection specifies: https://github.com/IQSS/dataverse/issues/756
- SWORD 2.0 Profile 6.4 "Retrieving the content" has not been implemented: https://github.com/IQSS/dataverse/issues/183
- Deaccessioning via API is not supported (it was in DVN 3.x): https://github.com/IQSS/dataverse/issues/778

- Let file metadata (i.e. description) be specified during zip upload: https://github.com/IQSS/dataverse/issues/723
- SWORD: Display of actual dcterms xml element for equivalent of required field not found: https://github.com/IQSS/dataverse/issues/1019

5.8.7 Bug fixes in v1.1

- Fix Abdera ArrayIndexOutOfBoundsException with non-existent atom-entry-study.xml in SWORD jar (upstream ideally) https://github.com/IQSS/dataverse/issues/893
- Sword API: Can't create study when hidden characters are introduced in atom.xml https://github.com/IQSS/dataverse/issues/894

5.8.8 Client libraries

- Python: https://github.com/swordapp/python-client-sword2
- Java: https://github.com/swordapp/JavaClient2.0
- R: https://github.com/IQSS/dataverse-client-r
- Ruby: https://github.com/swordapp/sword2ruby
- PHP: https://github.com/swordapp/swordappv2-php-library

5.9 Client Libraries

Listed below are a variety of clienty libraries to help you interact with Dataverse APIs from Python, R, Javascript, etc.

To get support for any of these client libraries, please consult each project's README.

Contents:

- *C/C++*
- *Go*
- *Java*
- *Javascript*
- *Julia*
- *PHP*
- *Python*
- *R*
- *Ruby*
- *Rust*

5.9.1 C/C++

https://github.com/aeonSolutions/OpenScience-Dataverse-API-C-library is the official C/C++ library for Dataverse APIs.

This C/C++ library was created and is currently maintained by Miguel T. To learn how to install and use it, see the project's wiki page.

5.9.2 Go

https://github.com/libis/rdm-dataverse-go-api is Go API library that can be used in your project by simply adding `github.com/libis/rdm-dataverse-go-api` as a dependency in your `go.mod` file. See the GitHub page for more details and usage examples.

5.9.3 Java

https://github.com/IQSS/dataverse-client-java is the official Java library for Dataverse APIs.

Richard Adams from ResearchSpace created and maintains this library.

5.9.4 Javascript

https://github.com/IQSS/dataverse-client-javascript is the official Javascript package for Dataverse APIs. It can be found on npm at https://www.npmjs.com/package/js-dataverse

It was created and is maintained by The Agile Monkeys.

5.9.5 Julia

https://github.com/gaelforget/Dataverse.jl is the official Julia package for Dataverse APIs. It can be found on JuliaHub (https://juliahub.com/ui/Packages/Dataverse/xWAqY/) and leverages pyDataverse to provide an interface to Dataverse's data access API and native API. Dataverse.jl provides a few additional functionalities with documentation (https://gaelforget.github.io/Dataverse.jl/dev/) and a demo notebook (https://gaelforget.github.io/Dataverse.jl/dev/notebook.html).

It was created and is maintained by Gael Forget.

5.9.6 PHP

There is no official PHP library for Dataverse APIs (please *get in touch* if you'd like to create one!) but there is a SWORD library written in PHP listed under *Client libraries* in the *SWORD API* documentation.

5.9.7 Python

There are multiple Python modules for interacting with Dataverse APIs.

EasyDataverse is a Python library designed to simplify the management of Dataverse datasets in an object-oriented way, giving users the ability to upload, download, and update datasets with ease. By utilizing metadata block configurations, EasyDataverse automatically generates Python objects that contain all the necessary details required to create the native Dataverse JSON format used to create or edit datasets. Adding files and directories is also possible with EasyDataverse and requires no additional API calls. This library is particularly well-suited for client applications such as workflows and scripts as it minimizes technical complexities and facilitates swift development.

python-dvuploader implements Jim Myers' excellent dv-uploader as a Python module. It offers parallel direct uploads to Dataverse backend storage, streams files directly instead of buffering them in memory, and supports multi-part uploads, chunking data accordingly.

pyDataverse primarily allows developers to manage Dataverse collections, datasets and datafiles. Its intention is to help with data migrations and DevOps activities such as testing and configuration management. The module is developed by Stefan Kasberger from AUSSDA - The Austrian Social Science Data Archive.

UBC's Dataverse Utilities are a set of Python console utilities which allow one to upload datasets from a tab-separated-value spreadsheet, bulk release multiple datasets, bulk delete unpublished datasets, quickly duplicate records. replace licenses, and more. For additional information see their PyPi page.

dataverse-client-python had its initial release in 2015. Robert Liebowitz created this library while at the Center for Open Science (COS) and the COS uses it to integrate the Open Science Framework (OSF) with Dataverse installations via an add-on which itself is open source and listed on the *Apps* page.

Pooch is a Python library that allows library and application developers to download data. Among other features, it takes care of various protocols, caching in OS-specific locations, checksum verification and adds optional features like progress bars or log messages. Among other popular repositories, Pooch supports Dataverse in the sense that you can reference Dataverse-hosted datasets by just a DOI and Pooch will determine the data repository type, query the Dataverse API for contained files and checksums, giving you an easy interface to download them.

idsc.dataverse reads metadata and files of datasets from a dataverse dataverse.example1.com and writes them into ~/.idsc/dataverse/api/dataverse.example1.com organized in directories PID_type/prefix/suffix, where PID_type is one of: hdl, doi or ark. It can then ''export" the local copy of the dataverse from ~/.idsc/dataverse/api/dataverse.example1.com to ~/.idsc/.cache/dataverse.example2.com so that one can upload them to dataverse.example2.com.

5.9.8 R

https://github.com/IQSS/dataverse-client-r is the official R package for Dataverse APIs. The latest release can be installed from CRAN. The R client can search and download datasets. It is useful when automatically (instead of manually) downloading data files as part of a script. For bulk edit and upload operations, we currently recommend pyDataverse.

The package is currently maintained by Shiro Kuriwaki. It was originally created by Thomas Leeper and then formerly maintained by Will Beasley.

5.9.9 Ruby

https://github.com/libis/dataverse_api is a Ruby gem for Dataverse APIs. It is registered as a library on Rubygems (https://rubygems.org/search?query=dataverse).

The gem is created and maintained by the LIBIS team (https://www.libis.be) at the University of Leuven (https://www.kuleuven.be).

5.9.10 Rust

https://github.com/gdcc/rust-dataverse

The Rust Dataverse client is a comprehensive crate designed for seamless interaction with the Dataverse API. It facilitates essential operations such as collection, dataset, and file management. Additionally, the crate includes a user-friendly command-line interface (CLI) that brings the full functionality of the library to the command line. This project is actively maintained by Jan Range.

5.10 Building External Tools

External tools can provide additional features that are not part of the Dataverse Software itself, such as data exploration. Thank you for your interest in building an external tool for the Dataverse Software!

Contents:

- *Introduction*
- *Examples of External Tools*
- *How External Tools Are Presented to Users*
- *Creating an External Tool Manifest*
  - *Examples of Manifests*
    - *External Tools for Files*
    - *External Tools for Datasets*
  - *Terminology*
  - *Reserved Words*
  - *Authorization Options*
    - *Signed URLs*
    - *API Token*
  - *Internationalization of Your External Tool*
  - *Using Example Manifests to Get Started*
- *Testing Your External Tool*
- *Spreading the Word About Your External Tool*
  - *Adding Your Tool to the Inventory of External Tools*
  - *Demoing Your External Tool*

- *Announcing Your External Tool*

5.10.1 Introduction

External tools are additional applications the user can access or open from your Dataverse installation to preview, explore, and manipulate data files and datasets. The term "external" is used to indicate that the tool is not part of the main Dataverse Software.

Note

Browser-based tools require CORS explicitly enabled in Dataverse. See *Cross-Origin Resource Sharing (CORS)* for details.

List every origin that will host your tool (or use * when a wildcard is acceptable and no authentication is required). If an origin is not listed, the browser will block that tool's API requests even if the tool page itself loads.

Once you have created the external tool itself (which is most of the work!), you need to teach a Dataverse installation how to construct URLs that your tool needs to operate. For example, if you've deployed your tool to *fabulousfiletool.com* your tool might want the ID of a file and the siteUrl of the Dataverse installation like this: *https://fabulousfiletool.com?fileId=42&siteUrl=https://demo.dataverse.org*

In short, you will be creating a manifest in JSON format that describes not only how to construct URLs for your tool, but also what types of files your tool operates on, where it should appear in the Dataverse installation web interfaces, etc.

The possibilities for external tools are endless. Let's look at some examples to get your creative juices flowing. Then we'll look at a complete list of parameters you can use when creating the manifest file for your tool.

If you're still looking for more information on external tools, you can also watch a video introduction called Background on the External Tool Framework (slides) from the 2020 Dataverse Community Meeting.

5.10.2 Examples of External Tools

Note: This is the same list that appears in the *External Tools* section of the Admin Guide.

Tool	Type	Sco	Description
Data Explorer	explore	file	A GUI which lists the variables in a tabular data file allowing searching, charting and cross tabulation analysis. The latest version incorporates the Data Curation Tool, a GUI for curating data by adding labels, groups, weights and other details to assist with informed reuse. See the README.md file at https://github.com/scholarsportal/Dataverse-Data-Explorer for the instructions on adding Data Explorer to your Dataverse.
Binder	explore	data	Binder allows you to spin up custom computing environments in the cloud (including Jupyter notebooks) with the files from your dataset. See https://github.com/IQSS/dataverse-binder-redirect for installation instructions.
File Previewers	explore	file	A set of tools that display the content of files - including audio, html, Hypothes.is annotations, images, PDF, Markdown, text, video, tabular data, spreadsheets, GeoJSON, zip, and NcML files - allowing them to be viewed without downloading the file. The previewers can be run directly from github.io, so the only required step is using the Dataverse API to register the ones you want to use. Documentation, including how to optionally brand the previewers, and an invitation to contribute through github are in the README.md file. Initial development was led by the Qualitative Data Repository and the spreasdheet previewer was added by the Social Sciences and Humanities Open Cloud (SSHOC) project. https://github.com/gdcc/dataverse-previewers
Ask the Data	query	file	Ask the Data is an experimental tool that allows you ask natural language questions about the data contained in Dataverse tables (tabular data). See the README.md file at https://github.com/IQSS/askdataverse/tree/main/askthedata for the instructions on adding Ask the Data to your Dataverse installation.
TurboCurator by ICPSR	configure	data	TurboCurator generates metadata improvements for title, description, and keywords. It relies on open AI's ChatGPT & ICPSR best practices. See the TurboCurator Dataverse Administrator page for more details on how it works and adding TurboCurator to your Dataverse installation.
JupyterHub	explore	file	The Dataverse-to-JupyterHub Data Transfer Connector is a tool that simplifies the transfer of data between Dataverse repositories and the cloud-based platform JupyterHub. It is designed for researchers, scientists, and data analysts, facilitating collaboration on projects by seamlessly moving datasets and files. The tool is a lightweight client-side web application built using React and relies on the Dataverse External Tool feature, allowing for easy deployment on modern integration systems. Currently optimized for small to medium-sized files, future plans include extending support for larger files and signed Dataverse endpoints. For more details, you can refer to the external tool manifest: https://forge.inrae.fr/dipso/eosc-pillar/dataverse-jupyterhub-connector/-/blob/master/externalTools.json
3DViewer by openforestdata.pl	explore	file	The 3DViewer by openforestdata.pl can be used to explore 3D files (e.g. STL format). It was presented by Kamil Guryn during the 2020 community meeting (slide deck, video) and can be found at https://github.com/OpenForestData/open-forest-data-previewers
Open OnDemand	explore	data	Open OnDemand is a web frontend to High Performance Computing (HPC) resources that is integrated with Dataverse. See https://iqss.github.io/ondemand-loop/ for installation instructions.

5.10.3 How External Tools Are Presented to Users

An external tool can appear in your Dataverse installation in a variety of ways:

- as an explore, preview, query or configure option for a file
- as an explore or configure option for a dataset
- as an embedded preview on the file landing page

See also the *Testing External Tools* section of the Admin Guide for some perspective on how Dataverse installations will expect to test your tool before announcing it to their users.

5.10.4 Creating an External Tool Manifest

External tools must be expressed in an external tool manifest file, a specific JSON format a Dataverse installation requires. As the author of an external tool, you are expected to provide this JSON file and installation instructions on a web page for your tool.

Examples of Manifests

Let's look at a few examples of external tool manifests (both at the file level and at the dataset level) before we dive into how they work.

External Tools for Files

`fabulousFileTool.json` is a file level (both an "explore" tool and a "preview" tool) that operates on tabular files:

```
{
  "displayName": "Fabulous File Tool",
  "description": "A non-existent tool that is fabulous fun for files!",
  "toolName": "fabulous",
  "scope": "file",
  "types": [
    "explore",
    "preview"
  ],
  "toolUrl": "https://fabulousfiletool.com",
  "contentType": "text/tab-separated-values",
  "httpMethod":"GET",
  "toolParameters": {
    "queryParameters": [
      {
        "fileid": "{fileId}"
      },
      {
        "datasetPid": "{datasetPid}"
      },
      {
        "locale":"{localeCode}"
      }
    ]
  },
```

(continues on next page)

(continued from previous page)

```
  "allowedApiCalls": [
    {
      "name":"retrieveDataFile",
      "httpMethod":"GET",
      "urlTemplate":"/api/v1/access/datafile/{fileId}",
      "timeOut":270
    }
  ]
}
```

`auxFileTool.json` is a file level preview tool that operates on auxiliary files associated with a data file (note the "requirements" section):

```
{
  "displayName": "AuxFileViewer",
  "description": "Show an auxiliary file from a dataset file.",
  "toolName": "auxPreviewer",
  "scope": "file",
  "types": [
    "preview"
  ],
  "toolUrl": "https://example.com/AuxFileViewer.html",
  "toolParameters": {
    "queryParameters": [
      {
        "fileid": "{fileId}"
      }
    ]
  },
  "requirements": {
    "auxFilesExist": [
      {
        "formatTag": "myFormatTag",
        "formatVersion": "0.1"
      }
    ]
  },
  "contentType": "application/foobar"
}
```

External Tools for Datasets

`dynamicDatasetTool.json` is a dataset level explore tool:

```
{
  "displayName": "Dynamic Dataset Tool",
  "description": "Dazzles! Dizzying!",
  "scope": "dataset",
  "types": [
    "explore"
  ],
```

(continues on next page)

(continued from previous page)

```
  "toolUrl": "https://dynamicdatasettool.com/v2",
  "toolParameters": {
    "queryParameters": [
      {
        "PID": "{datasetPid}"
      },
      {
        "locale":"{localeCode}"
      }
    ]
  },
  "allowedApiCalls": [
    {
      "name":"retrieveDatasetJson",
      "httpMethod":"GET",
      "urlTemplate":"/api/v1/datasets/{datasetId}",
      "timeOut":10
    }
  ]
}
```

Terminology

Term	Definition
external tool manifest	A **JSON file** the defines the URL constructed by a Dataverse installation when users click explore or configure tool options. External tool makers are asked to host this JSON file on a website (no app store yet, sorry) and explain how to use install and use the tool. Examples include `fabulousFileTool.json` and `dynamicDatasetTool.json` as well as the real world examples above such as Data Explorer.
displayName	The **name** of the tool in the Dataverse installation web interface. For example, "Data Explorer".
description	The **description** of the tool, which appears in a popup (for configure tools only) so the user who clicked the tool can learn about the tool before being redirected to the tool in a new tab in their browser. HTML is supported.
scope	Whether the external tool appears and operates at the **file** level or the **dataset** level. Note that a file level tool much also specify the type of file it operates on (see "contentType" below).
types	Whether the external tool is an **explore** tool, a **preview** tool, a **query** tool, a **configure** tool or any combination of these (multiple types are supported for a single tool). Configure tools require an API token because they make changes to data files (files within datasets). The older "type" keyword that allows you to pass a single type as a string is deprecated but still supported.
toolUrl	The **base URL** of the tool before query parameters are added.
contentType	File level tools operate on a specific **file type** (content type or MIME type such as "application/pdf") and this must be specified. Dataset level tools do not use contentType.
toolParameters	**httpMethod**, **queryParameters**, and **allowedApiCalls** are supported and described below.
httpMethod	Either `GET` or `POST`.
queryParameters	**Key/value combinations** that can be appended to the toolUrl. For example, once substitution takes place (described below) the user may be redirected to `https://fabulousfiletool.com?fileId=42&siteUrl=https://demo.dataverse.org`.
query parameter keys	An **arbitrary string** to associate with a value that is populated with a reserved word (described below). As the author of the tool, you have control over what "key" you would like to be passed to your tool. For example, if you want to have your tool receive and operate on the query parameter "dataverseFileId=42" instead of just "fileId=42", that's fine.
query parameter values	A **mechanism for substituting reserved words with dynamic content**. For example, in your manifest file, you can use a reserved word (described below) such as `{fileId}` to pass a file's database id to your tool in a query parameter. Your tool might receive this query parameter as "fileId=42".
reserved words	A **set of strings surrounded by curly braces** such as `{fileId}` or `{datasetId}` that will be inserted into query parameters. See the table below for a complete list.
allowedApiCalls	An array of objects defining callbacks the tool is allowed to make to the Dataverse API. If the dataset or file being accessed is not public, the callback URLs will be signed to allow the tool access for a defined time.
allowedApiCalls name	A name the tool will use to identify this callback URL such as `retrieveDataFile`.
allowedApiCalls urlTemplate	The relative URL for the callback using reserved words to indicate where values should by dynamically substituted such as `/api/v1/datasets/{datasetId}`.
allowedApiCalls httpMethod	Which HTTP method the specified callback uses such as `GET` or `POST`.
allowedApiCalls timeOut	For non-public datasets and datafiles, how many minutes the signed URLs given to the tool should be valid for. Must be an integer.
requirements	**Resources your tool needs to function.** For now, the only requirement you can specify is that one or more auxiliary files exist (see auxFilesExist in the *External Tools for Files* example). Currently, requirements only apply to preview tools. If the requirements are not met, the preview tool is not shown.
auxFilesExist	**An array containing formatTag and formatVersion pairs** for each auxiliary file that your tool needs to download to function properly. For example, a required aux file could have a `formatTag` of "NcML" and a `formatVersion` of "1.0". See also *Auxiliary File Support*.
toolName	A **name** of an external tool that is used to differentiate between external tools and also used in bundle.properties for localization in the Dataverse installation web interface. For example, the toolName for Data Explorer is `explorer`. For the Data Curation Tool the toolName is

Reserved Words

Reserved word	Status	Description
{siteUrl}	optional	The URL of the Dataverse installation from which the tool was launched. For example, https://demo.dataverse.org.
{fileId}	depends	The database ID of a file the user clicks "Explore" or "Configure" on. For example, 42. This reserved word is **required for file level tools** unless you use {filePid} instead.
{filePid}	depends	The Persistent ID (DOI or Handle) of a file the user clicks "Explore" or "Configure" on. For example, doi:10.7910/DVN/TJCLKP/3VSTKY. Note that not all Dataverse installations have Persistent IDs (PIDs) enabled at the file level. This reserved word is **required for file level tools** unless you use {fileId} instead.
{apiToken}	optional	The Dataverse installation's API token of the user launching the external tool, if available. Please note that API tokens should be treated with the same care as a password. For example, f3465b0c-f830-4bc7-879f-06c0745a5a5c.
{datasetId}	depends	The database ID of the dataset. For example, 42. This reseved word is **required for dataset level tools** unless you use {datasetPid} instead.
{datasetPid}	depends	The Persistent ID (DOI or Handle) of the dataset. For example, doi:10.7910/DVN/TJCLKP. This reseved word is **required for dataset level tools** unless you use {datasetId} instead.
{datasetVers:	optional	The friendly version number (or :draft) of the dataset version the file level tool is being launched from. For example, 1.0 or :draft.
{localeCode}	optional	The code for the language ("en" for English, "fr" for French, etc.) that user has selected from the language toggle in a Dataverse installation. See also *Internationalization*.

Authorization Options

When called for datasets or data files that are not public (i.e. in a draft dataset or for a restricted file), external tools are allowed access via the user's credentials. This is accomplished by one of two mechanisms:

Signed URLs

The signed URL mechanism is more secure than exposing API tokens and therefore recommended.

- Configured via the allowedApiCalls section of the manifest. The tool will be provided with signed URLs allowing the specified access to the given dataset or datafile for the specified amount of time. The tool will not be able to access any other datasets or files the user may have access to and will not be able to make calls other than those specified.
- For tools invoked via a GET call, Dataverse will include a callback query parameter with a Base64 encoded value. The decoded value is a signed URL that can be called to retrieve a JSON response containing all of the queryParameters and allowedApiCalls specified in the manfiest.
- For tools invoked via POST, Dataverse will send a JSON body including the requested queryParameters and allowedApiCalls. Dataverse expects the response to the POST to indicate a redirect which Dataverse will use to open the tool.

Note

An `allowedApiCalls` `urlTemplate` should not use any of Dataverse's reserved authentication-related query parameters `until`, `user`, `method`, `token`, `key` or `signed`: launching a tool whose manifest includes them will be rejected with an error naming the parameter. See *dataverse.api.signing-secret* for how the URLs are signed.

Note

For Dataverse site administrators: When Dataverse is behind a proxy, signed URLs may not work correctly due to protocol mismatches (HTTP vs HTTPS). Please refer to the *Using X-Forwarded-Proto for Signed URLs* section to ensure signed URLs work properly in proxy environments.

API Token

The API token mechanism is deprecated. Because it is less secure than signed URLs, it is not recommended for new external tools.

- Configured via the `queryParameters` by including an `{apiToken}` value. When this is present Dataverse will send the user's apiToken to the tool. With the user's API token, the tool can perform any action via the Dataverse API that the user could. External tools configured via this method should be assessed for their trustworthiness.
- For tools invoked via GET, this will be done via a query parameter in the request URL which could be cached in the browser's history. Dataverse expects the response to the POST to indicate a redirect which Dataverse will use to open the tool.
- For tools invoked via POST, Dataverse will send a JSON body including the apiToken.

Internationalization of Your External Tool

The name and description of your tool can be localized and made available in different languages in your Dataverse installation's web interface. Use the `toolName` parameter in the manifest JSON file and then add that toolName to bundle.properties.

For example, if the `toolName` of your external tool is `fabulous` then the lines in Bundle.properties should be:

```
externaltools.fabulous.displayname=Fabulous File Tool                externaltools.fabulous.
description=Fabulous Fun for Files!
```

Using Example Manifests to Get Started

Again, you can use `fabulousFileTool.json` or `dynamicDatasetTool.json` as a starting point for your own manifest file. Additional working examples, including ones using *Signed URLs*, are available at https://github.com/gdcc/dataverse-previewers .

5.10.5 Testing Your External Tool

As the author of an external tool, you are not expected to learn how to install and operate a Dataverse installation. There's a very good chance your tool can be added to a server Dataverse Community developers use for testing if you reach out on any of the channels listed under *Getting Help* in the Developer Guide.

By all means, if you'd like to install a Dataverse installation yourself, a number of developer-centric options are available. For example, there's a script to spin up a Dataverse installation on EC2 at https://github.com/gdcc/dataverse-ansible . The process for using curl to add your external tool to your Dataverse installation is documented under *Managing External Tools* in the Admin Guide.

5.10.6 Spreading the Word About Your External Tool

Adding Your Tool to the Inventory of External Tools

Once you've gotten your tool working, please make a pull request to update the list of tools above! You are also welcome to download `dataverse-external-tools.tsv`, add your tool to the TSV file, create and issue at https://github.com/IQSS/dataverse/issues , and then upload your TSV file there.

Unless your tool runs entirely in a browser, you may have integrated server-side software with your Dataverse installation. If so, please double check that your software is listed in the *Integrations* section of the Admin Guide and if not, please open an issue or pull request to add it. Thanks!

If you've thought to yourself that there ought to be an app store for Dataverse Software external tools, you're not alone. Please see https://github.com/IQSS/dataverse/issues/5688 :)

Demoing Your External Tool

https://demo.dataverse.org is the place to play around with the Dataverse Software and your tool can be included. Please email support@dataverse.org to start the conversation about adding your tool. Additionally, you are welcome to open an issue at https://github.com/gdcc/dataverse-ansible which already includes a number of the tools listed above.

Announcing Your External Tool

You are welcome to announce your external tool at https://groups.google.com/forum/#!forum/dataverse-community

If you're too shy, we'll do it for you. We'll probably tweet about it too. Thank you for your contribution to the Dataverse Project!

5.11 Dataset Curation Status API

When the *:AllowedCurationLabels* setting has been used to define Curation Labels, this API can be used to set these labels on draft datasets. Superusers can define which set of labels are allowed for a given datasets in a collection/an individual dataset using the api described in the *Managing Datasets and Dataverse Collections* section. The API here can be used by curators/those who have permission to publish the dataset to get/set/change/delete the label currently assigned to a draft dataset. If the *dataverse.ui.show-curation-status-to-all* flag is enabled, users who can see the draft dataset version can use the get API call.

This functionality is intended as a mechanism to integrate the Dataverse software with an external curation process/application: it is a way to make the state of a draft dataset, as defined in the external process, visible within Dataverse. These labels have no other effect in Dataverse and are only visible to curators/those with permission to publish the dataset. Any curation label assigned to a draft dataset will be removed upon publication.

Dataverse tracks the Curation Label as well as when it was assigned and by whom. It also keeps track of the history of prior assignments.

5.11.1 Get a Draft Dataset's Curation Status

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export DATASET_ID='12345'
export DATASET_PID='doi:10.5072/FK2A1B2C3'
export SERVER_URL=https://demo.dataverse.org

Example 1: Get the label using the DATASET ID

  curl -H X-Dataverse-key:$API_TOKEN "$SERVER_URL/api/datasets/$DATASET_ID/curationStatus
↪"

Example 2: Get the label using the DATASET PID

  curl -H X-Dataverse-key:$API_TOKEN "$SERVER_URL/api/datasets/:persistentId/
↪curationStatus?persistentId=$DATASET_PID"
```

You should expect a 200 ("OK") response and the draft dataset's curation status as a JSON object contained in a JSON 'data' object. The status will include a 'label','createTime', and the 'assigner'.

If the optional includeHistory query parameter is set to true, the responses 'data' entry will be a JSON array of curation status objects

curl -H X-Dataverse-key:$API_TOKEN "$SERVER_URL/api/datasets/:persistentId/curationStatus?persistentId=$DATASET_PID

For draft datasets that were created prior to v6.7, it is possible that curation status objects will have no createTime or assigner.

5.11.2 Set a Draft Dataset's Curation Label

To add a curation label for a draft Dataset, specify the Dataset ID (DATASET_ID) or Persistent identifier (DATASET_PID) and the label desired.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export DATASET_ID='12345'
export DATASET_PID='doi:10.5072/FK2A1B2C3'
export LABEL='Author contacted'
export SERVER_URL=https://demo.dataverse.org

Example: Add the label using the DATASET ID

  curl -X PUT -H X-Dataverse-key:$API_TOKEN "$SERVER_URL/api/datasets/$DATASET_ID/
↪curationStatus?label=$LABEL"

Example 2: Add a description using the DATASET PID

  curl -X PUT -H X-Dataverse-key:$API_TOKEN "$SERVER_URL/api/datasets/:persistentId/
↪curationStatus?label=$LABEL&persistentId=$DATASET_PID"
```

You should expect a 200 ("OK") response indicating that the label has been set. 403/Forbidden and 400/Bad Request responses are also possible, i.e. if you don't have permission to make this change or are trying to add a label that isn't in the allowed set or to add a label to a dataset with no draft version.

Note that Dataverse will add the current time as the createTime and the user as the 'assigner' of the label.

5.11.3 Delete a Draft Dataset's Curation Label

To delete the curation label on a draft Dataset, specify the Dataset ID (DATASET_ID) or Persistent identifier (DATASET_PID).

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export DATASET_PID='doi:10.5072/FK2A1B2C3'
export SERVER_URL=https://demo.dataverse.org

Example: Delete the label using the DATASET PID

  curl -X DELETE -H X-Dataverse-key:$API_TOKEN "$SERVER_URL/api/datasets/:persistentId/
↪curationStatus?persistentId=$DATASET_PID"
```

You should expect a 200 ("OK") response indicating the label has been removed.

5.11.4 Get the Set of Allowed Labels for a Dataset

To get the list of allowed curation labels allowed for a given Dataset

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export DATASET_ID='12345'
export DATASET_PID='doi:10.5072/FK2A1B2C3'
export SERVER_URL=https://demo.dataverse.org

Example 1: Get the label using the DATASET ID

  curl -H X-Dataverse-key:$API_TOKEN "$SERVER_URL/api/datasets/$DATASET_ID/
↪allowedCurationLabels"

Example 2: Get the label using the DATASET PID

  curl -H X-Dataverse-key:$API_TOKEN "$SERVER_URL/api/datasets/:persistentId/
↪allowedCurationLabels?persistentId=$DATASET_PID"
```

You should expect a 200 ("OK") response with a comma-separated list of allowed labels contained in a JSON 'data' object.

5.11.5 Get a Report on the Curation Status of All Datasets

To get a CSV file listing the curation statuses assigned to each Dataset with a draft version, along with the creation and last modification dates, and list of those with permissions to publish the version.

This API call is restricted to superusers.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org

Example: Get the report

  curl -H X-Dataverse-key:$API_TOKEN "$SERVER_URL/api/datasets/listCurationStates"
```

You should expect a 200 ("OK") response with a CSV formatted response.

The CSV response includes the following columns in order: #. Dataset Title (as a hyperlink to the dataset page) #. Creation Date of the draft dataset version #. Latest Modification Date of the draft dataset version #. Assigned curation status or '<none>' if no curation status is assigned but was previously, null if no curation state has every been set. #. Time when the curation status was applied to the draft dataset version #. The user who assigned this curation status #. (and beyond): Users (comma separated list) with the Roles (column headings) that can publish datasets and therefore see/set curation status When includeHistory is true, multiple rows may be present for each dataset, showing the full history of curation statuses.

5.12 Linked Data Notification API

Dataverse has an API implementing a Linked Data Notification (LDN) inbox allowing it to receive messages implementing the COAR Notify Relationship Announcement indicating a link between an external resource and a Dataverse dataset.

Dataverse has a related capability to send COAR Notify Relationship Announcement messages, automatically upon publication or manually. See the *Workflows* section of the Guides.

The motivating use case is to support a use case where Dataverse administrators may wish to create back-links to the remote resource (e.g. as a Related Publication, Related Material, etc.).

Upon receipt of a relevant message, Dataverse will create Announcement Received notifications for users who can edit the dataset involved. Notifications can be restricted to superusers who can publish the dataset as described below. (In the motivating use case, these superusers may then add an appropriate relationship and use the Update Curent Version publishing option to add it to the most recently published version of the dataset.)

The `dataverse.ldn.allowed-hosts` JVM option is a comma-separated list of hosts from which Dataverse will accept and process messages. By default, no hosts are allowed. * can be used in testing to indicate all hosts are allowed.

The `dataverse.ldn.coar-notify.relationship-announcement.notify-superusers-only` JVM option can be set to `true` to restrict notifications to superusers only (those who can publish the dataset). The default is to notify all users who can publish the dataset.

Messages can be sent via POST, using the application/ld+json ContentType:

```
export SERVER_URL=https://demo.dataverse.org

curl -X POST -H 'ContentType:application/ld+json' $SERVER_URL/api/inbox --upload-file␣
↪message.jsonld
```

The supported message format is described by the COAR Notify Relationship Announcement specification.

An example message is shown below. It indicates that a resource in the "Harvard DASH" test server has, as a "supplement", the dataset with DOI doi:10.5074/FKNOAHNQ. If this dataset is managed in the receiving Dataverse, a notification will be sent to user with the relevant permissions (as described above).


```
{
  "@context": [
    "https://www.w3.org/ns/activitystreams",
    "https://purl.org/coar/notify"
  ],
  "actor": {
    "id": "https://harvard-dash.staging.4science.cloud",
    "name": "Harvard DASH",
    "type": "Service"
  },
  "context": {
    "id": "https://harvard-dash.staging.4science.cloud/handle/1/42718322",
    "ietf:cite-as": "https://harvard-dash.staging.4science.cloud/handle/1/42718322",
    "ietf:item": {
      "id": "https://harvard-dash.staging.4science.cloud/bitstreams/e2ae80a1-35e5-411b-
↪9ef1-9175f6cccf23/download",
      "mediaType": "application/pdf",
      "type": [
        "Article",
        "sorg:ScholarlyArticle"
      ]
    },
    "type": "sorg:AboutPage"
  },
  "id": "urn:uuid:3c933c09-c246-473d-bea4-674db168cfee",
  "object": {
    "as:object": "doi: 10.5074/FKNOAHNQ",
    "as:relationship": "http://purl.org/vocab/frbr/core#supplement",
    "as:subject": "https://harvard-dash.staging.4science.cloud/handle/1/42718322",
    "id": "urn:uuid:0851f805-c52f-4d0b-81ac-a07e99c33e20",
    "type": "Relationship"
  },
  "origin": {
    "id": "https://harvard-dash.staging.4science.cloud",
    "inbox": "https://harvard-dash.staging.4science.cloud/server/ldn/inbox",
    "type": "Service"
  },
  "target": {
    "id": "http://ec2-3-238-245-253.compute-1.amazonaws.com/",
    "inbox": "http://ec2-3-238-245-253.compute-1.amazonaws.com/api/inbox",
    "type": "Service"
  },
  "type": [
    "Announce",
    "coar-notify:RelationshipAction"
  ]
}
```

5.13 Apps

The introduction of Dataverse Software APIs has fostered the development of a variety of software applications that are listed in the *Integrations*, *External Tools*, and *Reporting Tools and Common Queries* sections of the Admin Guide.

The apps below are open source and demonstrate how to use Dataverse Software APIs. Some of these apps are built on *Client Libraries* that are available for Dataverse Software APIs in Python, Javascript, R, Java, and Ruby.

Contents:

- *Javascript*
  - *Data Explorer*
  - *Data Curation Tool*
  - *File Previewers*
- *Python*
  - *dataverse-sample-data*
  - *Texas Digital Library dataverse-reports*
  - *OSF*
  - *dataverse-metrics*
  - *Whole Tale*
  - *Archivematica*
  - *repo2docker*
  - *dataverse-migration-scripts*
  - *idsc.dataverse*
  - *dataverse-metadata-crawler*
  - *mcp-dataverse*
- *Java*
  - *DVUploader*
  - *Dataverse for Android*
  - *DVImportExport*
- *Go*
  - *Integrations Dashboard*
- *PHP*
  - *DOI2PMH*
  - *OJS*
  - *OpenScholar*
- *Ruby*
  - *Open OnDemand Loop*

5.13.1 Javascript

Data Explorer

Data Explorer is a GUI which lists the variables in a tabular data file allowing searching, charting and cross tabulation analysis.

https://github.com/scholarsportal/Dataverse-Data-Explorer-v2

Data Curation Tool

Data Curation Tool is a GUI for curating data by adding labels, groups, weights and other details to assist with informed reuse.

https://github.com/scholarsportal/Dataverse-Data-Curation-Tool

File Previewers

File Previewers are tools that display the content of files - including audio, html, Hypothes.is annotations, images, PDF, text, video, GeoJSON - allowing them to be viewed without downloading.

https://github.com/gdcc/dataverse-previewers

5.13.2 Python

Please note that there are multiple Python modules for Dataverse Software APIs listed in the *Client Libraries* section.

dataverse-sample-data

dataverse-sample-data allows you to populate your Dataverse installation with sample data. It makes uses of pyDataverse, which is listed in the *Client Libraries* section.

https://github.com/IQSS/dataverse-sample-data

Texas Digital Library dataverse-reports

Dataverse Installation Reports from Texas Digital Library generates and emails statistical reports for a Dataverse installation using the native API and database queries.

https://github.com/TexasDigitalLibrary/dataverse-reports

OSF

OSF allows you to view, download, and upload files to and from a dataset in a Dataverse installation from an Open Science Framework (OSF) project.

https://github.com/CenterForOpenScience/osf.io/tree/develop/addons/dataverse

dataverse-metrics

dataverse-metrics aggregates and visualizes metrics across multiple Dataverse installations but can also be used with a single installation.

https://github.com/IQSS/dataverse-metrics

Whole Tale

Whole Tale enables researchers to analyze data using popular tools including Jupyter and RStudio with the ultimate goal of supporting publishing of reproducible research packages. As of 2025 the project is not active.

https://github.com/whole-tale/girder-wholetale/blob/v2.0.7/girder_wholetale/lib/dataverse/provider.py

Archivematica

Archivematica is an integrated suite of open-source tools for processing digital objects for long-term preservation.

https://github.com/artefactual/archivematica/tree/v1.9.2/src/MCPClient/lib/clientScripts

repo2docker

repo2docker is a command line tool that allows you to create and start a Docker image from a code repository that follows the [reproducible executable environment specification](https://repo2docker.readthedocs.io/en/latest/specification.html). repo2docker supports Dataverse installation DOIs to find and retrieve datasets.

https://github.com/jupyter/repo2docker/blob/master/repo2docker/contentproviders/dataverse.py

dataverse-migration-scripts

This series of Python scripts offers a starting point for migrating datasets from one Dataverse installation to another. Multiple parts of the process are handled in these scripts, including adding users, collections, and multiple versions of datasets. These scripts were developed to migrate data from version 4.20 to 5.1, but may provide a helpful starting point for other software versions. The *migration APIs* added in version 5.6 are not used. You can find more details in the repository, as well as this Google group thread.

https://github.com/scholarsportal/dataverse-migration-scripts

idsc.dataverse

This module can, among others, help you migrate one dataverse to another. (see migrate.md)

https://github.com/iza-institute-of-labor-economics/idsc.dataverse

dataverse-metadata-crawler

A Python CLI tool for bulk extraction of dataverses, datasets, and data file metadata from any chosen level of dataverse collection (an entire Dataverse repository/sub-Dataverse), with flexible export options to JSON and CSV formats.

https://github.com/scholarsportal/dataverse-metadata-crawler

mcp-dataverse

The code at https://github.com/gdcc/mcp-dataverse powers a *Model Context Protocol (MCP)* server for Dataverse.

5.13.3 Java

Please note that there is a Java library for Dataverse Software APIs listed in the *Client Libraries* section.

DVUploader

The open-source DVUploader tool is a stand-alone command-line Java application that uses the Dataverse Software API to upload files to a specified Dataset. Files can be specified by name, or the DVUploader can upload all files in a directory or recursively from a directory tree. The DVUploader can also verify that uploaded files match their local sources by comparing the local and remote fixity checksums. Source code, the latest release - jar file, and documentation are available on GitHub. DVUploader's creation was supported by the Texas Digital Library.

https://github.com/GlobalDataverseCommunityConsortium/dataverse-uploader

Dataverse for Android

Dataverse Software on Android makes use of a Dataverse installation's Search API.

https://github.com/IQSS/dataverse-android

DVImportExport

The open-source DVImportExport tool is a command-line tool for exporting/importing dataverses, datasets and files from one Dataverse instance to another. Source code, jar files and documentation are available on Gitlab. DVImport-Export is developed at the Luxembourg Centre for Contemporary and Digital History of the Université de Luxembourg.

https://gitlab.com/uniluxembourg/c2dh/dataverse/dataverse-import-export

5.13.4 Go

Integrations Dashboard

The integrations dashboard is software by the Dataverse community to enable easy data transfer from an existing data management platform to a dataset in a Dataverse collection. See *Integrations Dashboard* for details.

https://github.com/libis/rdm-integration

5.13.5 PHP

DOI2PMH

The DOI2PMH server allow Dataverse instances to harvest DOI through OAI-PMH from otherwise unharvestable sources.

https://github.com/IQSS/doi2pmh-server

OJS

The Open Journal Systems (OJS) Dataverse Plugin adds data sharing and preservation to the OJS publication process.

https://github.com/lepidus/dataversePlugin

OpenScholar

The Dataverse Software module from OpenScholar allows a Dataverse installation's widgets to be easily embedded in its web pages:

https://github.com/openscholar/d7-openscholar/tree/SCHOLAR-3.x/openscholar/modules/os_features/os_dataverse

5.13.6 Ruby

Open OnDemand Loop

Open OnDemand Loop integrates Dataverse with Open OnDemand, as *described* in the Admin Guide.

https://github.com/IQSS/ondemand-loop

5.14 Frequently Asked Questions

APIs are less intuitive than graphical user interfaces (GUIs) so questions are expected!

Contents:

- *What is an API?*
- *What Are Common Use Cases for Dataverse Software APIs?*
- *Where Can I Find Examples of Using Dataverse Software APIs?*
- *When Should I Use the Native API vs. the SWORD API?*
- *To Operate on a Dataset Should I Use Its DOI (or Handle) or Its Database ID?*
- *Where is the Comprehensive List of All API Functionality?*
- *Is There a Changelog of API Functionality That Has Been Added Over Time?*
- *What Functionality is GUI Only and Not Available Via API*
- *Why Are the Return Values (HTTP Status Codes) Not Documented?*

- *What If My Question Is Not Answered Here?*

5.14.1 What is an API?

See "What is an API?" in the *Introduction* section.

5.14.2 What Are Common Use Cases for Dataverse Software APIs?

See the *Getting Started with APIs* section for common use cases for researchers and curators. Other types of API users should find starting points at *Types of Dataverse Software API Users*.

5.14.3 Where Can I Find Examples of Using Dataverse Software APIs?

See the *Getting Started with APIs* section links to examples using curl.

For examples in Javascript, Python, R, and Java, and PHP, see the *Apps* and *Client Libraries* sections.

5.14.4 When Should I Use the Native API vs. the SWORD API?

The *SWORD API* is based on a standard, works fine, and is fully supported, but much more development effort has been going into the *Native API*, which is not based on a standard. It is specific to the Dataverse Software.

SWORD uses XML. The Native API uses JSON.

SWORD only supports a dozen or so operations. The Native API supports many more.

5.14.5 To Operate on a Dataset Should I Use Its DOI (or Handle) or Its Database ID?

It is fine to target a dataset using either its Persistent ID (PID such as DOI or Handle) or its database id.

Here's an example from *Update Dataset Terms of Access* of targeting a dataset using its DOI:

```
curl -H X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx -X POST "https://demo.
↪dataverse.org/api/datasets/:persistentId/actions/:publish?persistentId=doi:10.5072/FK2/
↪J8SJZB&type=major"
```

You can target the same dataset with its database ID ("42" in the example below), like this:

```
curl -H X-Dataverse-key:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx -X POST "https://demo.
↪dataverse.org/api/datasets/42/actions/:publish?type=major"
```

Note that when multiple query parameters are used (such as `persistentId` and `type` above) there is a question mark (?) before the first query parameter and ampersands (&) before each of the subsequent query parameters. Also, & has special meaning in Unix shells such as Bash so you must put quotes around the entire URL.

5.14.6 Where is the Comprehensive List of All API Functionality?

There are so many Dataverse Software APIs that a single page in this guide would probably be overwhelming. See *Lists of Dataverse APIs* for links to various pages.

It is possible to get a complete list of API functionality in Swagger/OpenAPI format. See *Getting the OpenAPI Document*.

5.14.7 Is There a Changelog of API Functionality That Has Been Added Over Time?

Changes to the API that don't break anything can be found in the release notes of each release. Breaking changes are documented in *API Changelog (Breaking Changes)*.

5.14.8 What Functionality is GUI Only and Not Available Via API

The following tasks cannot currently be automated via API because no API exists for them. The web interface should be used instead for these GUI-only features:

- Setting a logo image, URL, and tagline when creating a Dataverse collection.
- Editing properties of an existing Dataverse collection.
- Set "Enable Access Request" for Terms of Use: https://groups.google.com/d/msg/dataverse-community/oKdesT9rFGc/qM6wrsnnBAAJ
- Downloading a guestbook.
- Set guestbook_id for a dataset: https://groups.google.com/d/msg/dataverse-community/oKdesT9rFGc/qM6wrsnnBAAJ
- Filling out a guestbook. See also https://groups.google.com/d/msg/dataverse-dev/G9FNGP_bT0w/dgE2Fk4iBQAJ
- Seeing why a file failed ingest.
- Dataset templates.
- Deaccessioning datasets.

If you would like APIs for any of the features above, please open a GitHub issue at https://github.com/IQSS/dataverse/issues

You are also welcome to open an issue to add to the list above. Or you are welcome to make a pull request. Please see the *Writing Documentation* section of the Contributor Guide for instructions.

5.14.9 Why Are the Return Values (HTTP Status Codes) Not Documented?

They should be. Please consider making a pull request to help. The *Writing Documentation* section of the Contriburor Guide should help you get started. *Create a Dataverse Collection* has an example you can follow or you can come up with a better way.

Also, please note that we are starting to experiment with putting response codes in our OpenAPI document. See *Getting the OpenAPI Document*.

5.14.10 What If My Question Is Not Answered Here?

Please ask! For information on where to ask, please see *Getting Help*.

5.15 API Changelog (Breaking Changes)

This API changelog is experimental and we would love feedback on its usefulness. Its primary purpose is to inform API developers of any breaking changes. (We try not to ship any backward incompatible changes, but it happens.) To see a list of new APIs and backward-compatible changes to existing API, please see each version's release notes at https://github.com/IQSS/dataverse/releases

Contents:

- *v6.12*
- *v6.11*
- *v6.10*
- *v6.9*
- *v6.8*
- *v6.7*
- *v6.6*
- *v6.5*
- *v6.4*
- *v6.3*
- *v6.2*
- *v6.1*
- *v6.0*
- *v5.6*

5.15.1 v6.12

- Several API endpoints that return both a `message` and `data` field were incorrectly returning the message as a nested object (`{"message":{"message":"..."}}`). This has been fixed so that the message is now a plain string (`{"message":"..."}`). If you have integrations that depend on the old behavior, you can temporarily revert by setting *dataverse.legacy.api-response-message-style* to true. This flag will be removed in a future version. Affected endpoints: `POST /api/datasets/{id}/add` (duplicate file warning), `PUT /api/admin/settings`, `PUT /api/dataverses/{id}`, `PUT /api/dataverses/{id}/inputLevels`, `POST /api/admin/savedsearches`, `PUT /api/harvest/clients/{nickName}`, `PUT /api/harvest/server/oaisets/{specname}`. See #12096.
- Most API endpoints that return a success notification but no actual data have it embedded into `data`: `{"data":{"message":"..."}}`. For now, this style will remain the supported default. In a future version of Dataverse the `message` will always be a separate top field: `{"data":{},"message":"..."}`. Integrators

and client vendors are welcome to opt-in to the new style and test thoroughly by enabling *dataverse.feature.unify-api-response-message-style*.

- The permission reindexing endpoints have been updated to use `POST` and require superuser access. They are now documented in the *Solr Search Index* guide.
- The datafile/{fileId}/metadata/ddi (see *Data Variable Metadata Access*) now returns 403/Forbidden rather than 400/Bad Request when the caller can't access the file due to permissions, embargo, or retention period (consistent with other file access APIs)
  - **/api/admin/index/perms**
  - **/api/admin/index/perms/{id}**
- Dataset creation API calls may now behave differently when neither a license nor terms are provided, depending on the new *dataverse.feature.do-not-assume-default-license* feature flag.
- Whether file extensions are included in the "Content-disposition" header returned when downloading auxiliary files depends on whether the relevant format is one of the content types supported in Tika. A recent update to the version of Tika has added new content types, including "text/markdown", and auxiliary files with these types now have a a file extension included (e.g. ".md" in this case) in the header.

5.15.2 v6.11

- The GET /api/mydata/retrieve, if the search returns no data, now includes the "data" block with 0 results. The message that was returned in "error_message" will be returned in "message" and the "success" will be *true*. All other errors will continue to reply with "success":false and the error message in "error_message".
- The endpoints GET, PUT AND DELETE for */api/admin/dataverse/{alias}/storageDriver* have been moved to */api/dataverses/{alias}/storageDriver*.
- The endpoint */api/admin/dataverse/storageDrivers* has been moved and renamed to */api/dataverses/{alias}/allowedStorageDrivers*. Regarding the change of the name, this endpoint will in the future only display the storageDrivers that are allowed on the specified collection, as of now, it will display the entire list of available Drivers on the installation.
- The following API will now return `403` if the `requireFilesToPublishDataset` flag is set and the dataset version contains 0 files.
  - **/api/datasets/{Id}/submitForReview**
- The `GET /api/access/datafile/{id}/userPermissions` endpoint now requires authentication.
- The Croissant *metadata export format* has been updated from version 1.0 to 1.1, which is reflected in the `conformsTo` property. The unused `wd` property has been dropped.

5.15.3 v6.10

- The following GET APIs will now return `400` if a required Guestbook Response is not supplied. A Guestbook Response can be passed to these APIs in the JSON body using a POST call. See the notes under *Basic File Access* and *Download By Dataset By Version* for details.
  - **/api/access/datafile/{fileId:.+}**
  - **/api/access/datafiles/{fileIds}**
  - **/api/access/dataset/{id}**
  - **/api/access/dataset/{id}/versions/{versionId}**

- The following POST APIs will now return `400` if a required Guestbook Response is not supplied. A Guestbook Response can be passed to these APIs in the JSON body. See the note under *Basic Download By Dataset* for details.
  - **/api/access/datafiles**
  - **/api/access/datafile/bundle/{fileId}**
- The following PUT APIs will now return `400` if a required Guestbook Response is not supplied. When JVM setting -Ddataverse.files.guestbook-at-request=true is set a Guestbook Response may be required to be passed to these APIs in the JSON body. See the note under Configuration *dataverse.files.guestbook-at-request* for details.
  - **/api/access/datafile/{id}/requestAccess**

5.15.4 v6.9

- When creating datasets that contain a datasetType, that datasetType must be allowed at the collection level. This can be accomplished by passing `allowedDatasetTypes` to the *Change Collection Attributes* API.
- The POST /api/admin/makeDataCount/{id}/updateCitationsForDataset processing is now asynchronous and the response no longer includes the number of citations. The response can be OK if the request is queued or 503 if the queue is full (default queue size is 1000).
- The way to set per-format size limits for tabular ingest has changed. JSON input is now used. See *:TabularIngestSizeLimit*.
- In the past, the settings API would accept any key and value. This is no longer the case because validation has been added. See *Configure Single Database Setting*, for example.
- For GET /api/notifications/all the JSON response has changed breaking the backward compatibility of the API.
- For GET /api/admin/dataverse/{dataverse-alias}/storageDriver and /api/datasets/{identifier}/storageDriver the driver name is no longer returned in data.message. Instead, it is returned as data.name (along with other information about the storageDriver).

5.15.5 v6.8

- For POST /api/files/{id}/metadata passing an empty string ("description":"") or array ("categories":[]) will no longer be ignored. Empty fields will now clear out the values in the file's metadata. To ignore the fields simply do not include them in the JSON string.
- For PUT /api/datasets/{id}/editMetadata the query parameter "sourceInternalVersionNumber" has been removed and replaced with "sourceLastUpdateTime" to verify that the data being edited hasn't been modified and isn't stale.
- For GET /api/dataverses/$dataverse-alias/links the JSON response has changed breaking the backward compatibility of the API.
- For PUT /api/dataverses/$dataverse-alias/inputLevels custom input levels that had been previously set will no longer be deleted. To delete input levels send an empty list (deletes all), then send the new/modified list.
- For GET /api/externalTools and /api/externalTools/{id} the responses are now formatted as JSON (previously the toolParameters and allowedApiCalls were a JSON object and array (respectively) that were serialized as JSON strings) and any configured "requirements" are included.

5.15.6 v6.7

- An undocumented *Search API* parameter called "show_my_data" has been removed. It was never exercised by tests and is believed to be unused. API users should use the *MyData* API instead.
- /api/datasets/{id}/curationStatus API now includes a JSON object with curation label, createtime, and assigner rather than a string 'label' and it supports a new boolean includeHistory parameter (default false) that returns a JSON array of statuses
- /api/datasets/{id}/listCurationStates includes new columns "Status Set Time" and "Status Set By" columns listing the time the current status was applied and by whom. It also supports the boolean includeHistory parameter.
- Due to updates in libraries used by Dataverse, XML serialization may have changed slightly with respect to whether self-closing tags are used for empty elements. This primarily affects XML-based metadata exports. The XML structure of the export itself has not changed, so this is only an incompatibility if you are not using an XML parser.

5.15.7 v6.6

- The JSON representation for a datasetVersion sent or received in API calls has changed such that
  - "versionNote" -> "deaccessionNote"
  - "archiveNote" –> "deaccessionLink"
  - These may be non-null for deaccessioned versions and an optional new "versionNote" field indicating the reason a version was created may be present on any datasetversion.
- **/api/metadatablocks** is no longer returning duplicated metadata properties and does not omit metadata properties when called.
- **/api/roles**: *Show Role* now properly returns 403 Forbidden instead of 401 Unauthorized when you pass a working API token that doesn't have the right permission.
- The content type for the `schema.org` dataset metadata export format has been corrected. It was `application/json` and now it is `application/ld+json`. See also *Export Metadata of a Dataset in Various Formats*.

5.15.8 v6.5

- **/api/datasets/{identifier}/links**: The response from *List Collections that are Linked from a Dataset* has been improved to provide a more structured (but backward-incompatible) JSON response.

5.15.9 v6.4

- **/api/datasets/$dataset-id/modifyRegistration**: Changed from GET to POST
- **/api/datasets/modifyRegistrationPIDMetadataAll**: Changed from GET to POST

5.15.10 v6.3

- **/api/admin/superuser/{identifier}**: The POST endpoint that toggles superuser status has been deprecated in favor of a new PUT endpoint that allows you to specify true or false. See *Set Superuser Status*.

5.15.11 v6.2

- The fields "northLongitude" and "southLongitude" have been deprecated in favor of "northLatitude" and "southLatitude" in the Geolocation metadata block. After upgrading to 6.2 or later, you will need to use the new fields when creating or updating a dataset.
- **/api/datasets/{id}/versions/{versionId}**: The includeFiles parameter has been renamed to excludeFiles. The default behavior remains the same, which is to include files. However, when excludeFiles is set to true, the files will be excluded. A bug that caused the API to only return a deaccessioned dataset if the user had edit privileges has been fixed.
- **/api/datasets/{id}/versions**: The includeFiles parameter has been renamed to excludeFiles. The default behavior remains the same, which is to include files. However, when excludeFiles is set to true, the files will be excluded.
- **/api/files/$ID/uningest**: Can now be used by users with the ability to publish the dataset to undo a failed ingest. (Removing a successful ingest still requires being superuser)

5.15.12 v6.1

- The metadata field "Alternative Title" now supports multiple values so you must pass an array rather than a string when populating that field via API. See https://github.com/IQSS/dataverse/pull/9440

5.15.13 v6.0

- **/api/access/datafile**: When a null or invalid API token is provided to download a public (non-restricted) file with this API call, it will result on a `401` error response. Previously, the download was allowed (`200` response). Please note that we noticed this change sometime between 5.9 and 6.0. If you can help us pinpoint the exact version (or commit!), please get in touch. See *Data Access API*.
- **/openapi**: This endpoint is currently broken. See https://github.com/IQSS/dataverse/issues/9981

5.15.14 v5.6

- **/api/dataverses/$PARENT/datasets**: The "create dataset" API endpoint now requires the header `Content-type:application/json` to be passed. The error can be confusing, saying something about validation, such as `'{"status":"ERROR","message":"Validation Failed: Title is required. (Invalid value:edu.harvard.iq.dataverse.DatasetField[ id=null ])....` See *Create a Dataset in a Dataverse Collection*.

CHAPTER

SIX

INSTALLATION GUIDE

Contents:

6.1 Introduction

Welcome! Thanks for installing The Dataverse Project!

Contents:

- *Quick Links*
- *Intended Audience*
- *Related Guides*
- *Getting Help*
  - *Information to Send to Support When Installation Fails*
- *Improving this Guide*

6.1.1 Quick Links

If you are installing the Dataverse Software for the first time, please proceed to the *Preparation* section.

Jump ahead to *Configuration* or *Upgrading* for an existing Dataverse installation.

6.1.2 Intended Audience

This guide is intended primarily for sysadmins who are installing, configuring, and upgrading a Dataverse installation. This guide was written with non-Docker installation in mind but if you're interested in Docker, see the *Container Guide*.

Sysadmins are expected to be comfortable using standard Linux commands, issuing `curl` commands, and running SQL scripts.

6.1.3 Related Guides

See the *Container Guide* if you want to run Dataverse in Docker.

Many "admin" functions can be performed by Dataverse installation users themselves (non-superusers) as documented in the *User Guide* and that guide is a good introduction to the features of the Dataverse Software from an end user perspective.

If you are a sysadmin who likes to code, you may find the *API Guide* useful, and you may want to consider improving the installation script or hacking on the community-lead configuration management options mentioned in the *Preparation* section. If you **really** like to code and want to help with the Dataverse code or documentation, please check out the *Contributor Guide* and the *Developer Guide*!

6.1.4 Getting Help

To get help installing or configuring a Dataverse installation, please try one or more of:

- posting to the dataverse-community Google Group.
- asking at https://chat.dataverse.org
- emailing support@dataverse.org to open a private ticket at https://help.hmdc.harvard.edu

Information to Send to Support When Installation Fails

If you've encountered a problem installing Dataverse and are ready to ask for help, please consider sending along the following information so that the Dataverse team and community can more easily assist you.

- Version of Dataverse you are trying to install.
- Operating system (usually a Linux distribution) and version.
- Output from the installer (STDOUT, STDERR).
- The `scripts/api/setup-all.*.log` files left behind by the installer.
- The `server.log` file from Payara (by default at `/usr/local/payara7/glassfish/domains/domain1/logs/server.log`).

6.1.5 Improving this Guide

If you spot a typo in this guide or would like to suggest an improvement, please find the appropriate file in https://github.com/IQSS/dataverse/tree/develop/doc/sphinx-guides/source/installation and send a pull request as explained in the *Writing Documentation* section of the Contributor Guide. You are also welcome to simply open an issue at https://github.com/IQSS/dataverse/issues to describe the problem with this guide.

Next is the *Preparation* section.

6.2 Preparation

```
> "What are you preparing? You're always preparing! Just go!" -- Spaceballs
```

We'll try to get you up and running as quickly as possible, but we thought you might like to hear about your options. :)

Contents:

- *Choose Your Own Installation Adventure*
  - *Standard Installation*
  - *Docker Installation*
  - *Advanced Installation*
- *Architecture and Components*
  - *Required Components*
  - *Optional Components*
- *System Requirements*
  - *Hardware Requirements*
  - *Software Requirements*
- *Decisions to Make*
- *Next Steps*

6.2.1 Choose Your Own Installation Adventure

Standard Installation

Installing the Dataverse Software involves some system configuration followed by executing an installation script that will guide you through the installation process as described in *Installation*, but reading about the *Architecture and Components* of the Dataverse Software is recommended first.

Docker Installation

See the *Container Guide*, especially *Running Dataverse in Docker* and *Production*.

Advanced Installation

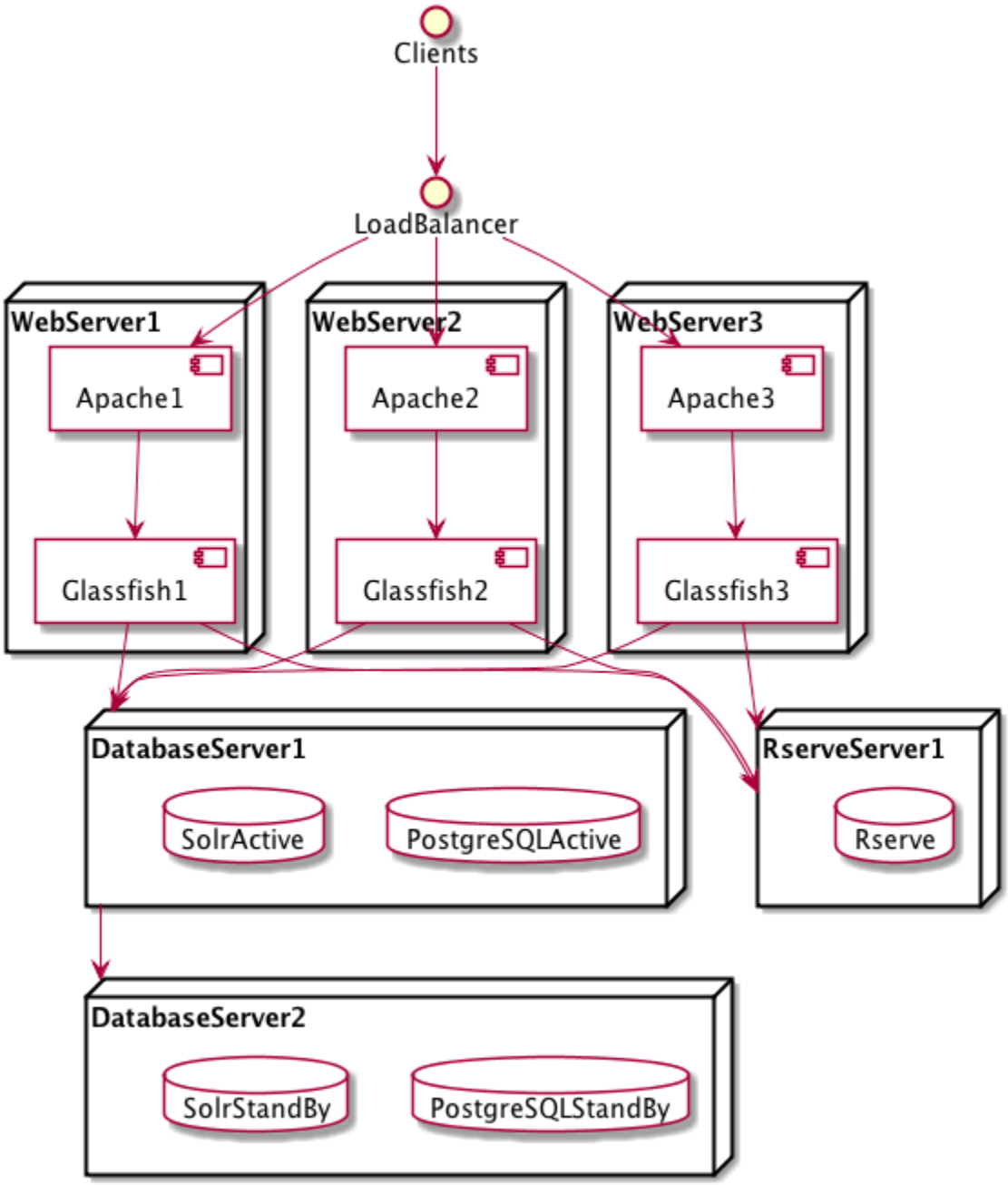
There are some community-lead projects to use configuration management tools such as Ansible and Puppet to automate the installation and configuration of the Dataverse Software, but support for these solutions is limited to what the Dataverse Community can offer as described in each project's webpage:

- https://github.com/gdcc/dataverse-ansible
- https://gitlab.com/lip-computing/dataverse
- https://github.com/IQSS/dataverse-puppet

(Please note that the "dataverse-ansible" repo is used in a script that allows the Dataverse Software to be installed on Amazon Web Services (AWS) from arbitrary GitHub branches as described in the *Deployment* section of the Developer Guide.)

The Dataverse Project team is happy to "bless" additional community efforts along these lines (i.e. Chef, Salt, etc.) by creating a repo under https://github.com/gdcc and managing team access.

The Dataverse Software permits a fair amount of flexibility in where you choose to install the various components. The diagram below shows a load balancer, multiple proxies and web servers, redundant database servers, and offloading of potentially resource intensive work to a separate server. (Glassfish is shown rather than Payara.)



A setup such as this is advanced enough to be considered out of scope for this guide (apart from a stub in the *Advanced Installation* section) but you are welcome to ask questions about similar configurations via the support channels listed in the *Introduction*.

6.2.2 Architecture and Components

The Dataverse Software is a Java Enterprise Edition (EE) web application that is shipped as a WAR (Web ARchive) file. This WAR file is deployed to an application server or app server.

Required Components

When planning your installation you should be aware of the following components of the Dataverse Software architecture:

- Linux: RHEL or derivative is highly recommended since all development and QA happens on this distribution.
- App server: Payara is the recommended Jakarta EE application server.
- PostgreSQL: a relational database.
- Solr: a search engine. A Dataverse Software-specific schema is provided.
- SMTP server: for sending mail for password resets and other notifications.
- Persistent identifier service: DOI and Handle support are provided. Production use requires a registered DOI or Handle.net authority.
- Rserve: runs as a daemon to execute R code.

Optional Components

There are a number of optional components you may choose to install or configure, including:

- External Tools: Third party tools for data exploration can be added to the Dataverse installation by following the instructions in the *External Tools* section of the Admin Guide.
- Dropbox integration *dataverse.dropbox.key*: for uploading files from the Dropbox API.
- Apache: a web server that can "reverse proxy" Jakarta EE applications (like the Dataverse Software) and rewrite HTTP traffic.
- Shibboleth: an authentication system described in *Shibboleth*. Its use with a Dataverse installation requires Apache.
- OAuth2: an authentication system described in *OAuth Login Options*.

See also the *Integrations* section of the Admin Guide.

6.2.3 System Requirements

Hardware Requirements

A basic Dataverse installation runs fine on modest hardware. For example, in the recent past we had a test instance backed by a single virtual machine with two 2.8 GHz processors, 8 GB of RAM and 50 GB of disk.

In contrast, before we moved it to the Amazon Cloud, the production installation at https://dataverse.harvard.edu was backed by six servers with two Intel Xeon 2.53 Ghz CPUs and either 48 or 64 GB of RAM. The three servers with 48 GB of RAM run were web frontends running Glassfish 4 and Apache and were load balanced by a hardware device. The remaining three servers with 64 GB of RAM were the primary and backup database servers and a server dedicated to running Rserve. Multiple TB of storage were mounted from a SAN via NFS.

Currently, the Harvard Dataverse Repository is served by four AWS server nodes

- two instances for web frontends running Payara fronted by Apache ("m4.4xlarge" with 64 GB RAM and 16 vCPUs)
  - these are sitting behind an AWS ELB load balancer
- one instance for the Solr search engine ("m4.2xlarge" with 32 GB RAM and 8 vCPUs)
- one instance for R ("m4.xlarge" instances with 16 GB RAM and 4 vCPUs)

The PostgreSQL database is served by Amazon RDS.

Physical files are stored on Amazon S3. The primary bucket is replicated in real-time to a secondary bucket, which is backed up to Glacier. Deleted files are kept around on the secondary bucket for a little while for convenient recovery. In addition, we use a backup script mentioned under *Backups*.

Experimentation and testing with various hardware configurations is encouraged, or course. Note that the installation script will attempt to give your app server (the web frontend) the right amount of RAM based on your system.

Software Requirements

See *Architecture and Components* for an overview of required and optional components. The *Prerequisites* section is oriented toward installing the software necessary to successfully run the Dataverse Software installation script. Pages on optional components contain more detail of software requirements for each component.

Clients are expected to be running a relatively modern browser.

6.2.4 Decisions to Make

Here are some questions to keep in the back of your mind as you test and move into production:

- How much storage do I need? What is the scale of data I will need to handle (see *Scaling Dataverse with Data Size*)?
- Which features do I want based on *Architecture and Components*?
- How do I want my users to log in to the Dataverse installation? With local accounts? With Shibboleth/SAML? With OAuth providers such as ORCID, GitHub, or Google?
- Do I want to to run my app server on the standard web ports (80 and 443) or do I want to "front" my app server with a proxy such as Apache or nginx? See "Network Ports" in the *Configuration* section.
- How many points of failure am I willing to tolerate? How much complexity do I want?
- How much does it cost to subscribe to a service to create persistent identifiers such as DOIs or handles?
- What licenses should I make available to my users?

6.2.5 Next Steps

Proceed to the *Prerequisites* section which will help you get ready to run the Dataverse Software installation script.

6.3 Prerequisites

Before running the Dataverse Software installation script, you must install and configure Linux, Java, Payara, PostgreSQL, Solr, and jq. The other software listed below is optional but can provide useful features.

After following all the steps below, you can proceed to the *Installation* section.

Contents:

- *Linux*
- *Java*
  - *Installing Java*
- *Payara*
  - *Installing Payara*
  - *Launching Payara on System Boot*
- *PostgreSQL*
  - *Installing PostgreSQL*
  - *Configuring Database Access for the Dataverse Installation (and the Dataverse Software Installer)*
- *Solr*
  - *Supported Versions*
  - *Installing Solr*
  - *Solr Init Script*
  - *Securing Solr*
- *jq*
  - *Installing jq*
- *ImageMagick*
  - *Installing and configuring ImageMagick*
- *R*
  - *Installing R*
  - *Installing the required R libraries*
  - *Rserve*
- *Counter Processor*
  - *Installing Counter Processor*
  - *Installing GeoLite Country Database*
  - *Creating a counter User*
  - *Installing Counter Processor Python Requirements*
- *Next Steps*

6.3.1 Linux

We assume you plan to run your Dataverse installation on Linux and we recommend RHEL or a derivative such as Rocky Linux, which is the distribution family tested by the Dataverse Project team. These instructions are written for RHEL9 and derivatives, but Dataverse is known to work well in Debian, Ubuntu, and most any modern Linux distribution.

6.3.2 Java

The recommended version is Java 21 because it's the version we test with.

Installing Java

The Dataverse Software should run fine with only the Java Runtime Environment (JRE) installed, but installing the Java Development Kit (JDK) is recommended so that useful tools for troubleshooting production environments are available. We recommend using Oracle JDK or OpenJDK.

The Oracle JDK can be downloaded from https://www.oracle.com/technetwork/java/javase/downloads/index.html

On a RHEL/derivative OS, install OpenJDK (devel version) using dnf:

```
# sudo dnf install java-21-openjdk
```

If you have multiple versions of Java installed, Java 21 should be the default when `java` is invoked from the command line. You can test this by running `java -version`.

On RHEL/derivative you can make Java 21 the default with the `alternatives` command, having it prompt you to select the version of Java from a list:

```
# alternatives --config java
```

6.3.3 Payara

Payara 7.2026.9 is recommended. Newer versions might work fine. Regular updates are recommended.

Installing Payara

Note: The Dataverse Software installer need not be run as root, and it is recommended that Payara not run as root either. We suggest the creation of a "dataverse" service account for this purpose:

```
# useradd dataverse
```

- Download and install Payara (installed in `/usr/local/payara7` in the example commands below):

```
# wget https://nexus.payara.fish/repository/payara-community/fish/payara/
↪distributions/payara/7.2026.9/payara-7.2026.9.zip
# unzip payara-7.2026.9.zip
# mv payara7 /usr/local
```

If nexus.payara.fish is ever down for maintenance, Payara distributions are also available from https://repo1.maven.org/maven2/fish/payara/distributions/payara/

If you intend to install and run Payara under a service account (and we hope you do), chown -R the Payara hierarchy to root to protect it but give the service account access to the below directories:

- Set service account permissions:

```
# chown -R root:root /usr/local/payara7
# chown dataverse /usr/local/payara7/glassfish/lib
# chown -R dataverse:dataverse /usr/local/payara7/glassfish/domains/domain1
```

After installation, you may chown the lib/ directory back to root; the installer only needs write access to copy the JDBC driver into that directory.

- Change from `-client` to `-server` under `<jvm-options>-client</jvm-options>`:

```
# vim /usr/local/payara7/glassfish/domains/domain1/config/domain.xml
```

This recommendation comes from http://www.c2b2.co.uk/middleware-blog/glassfish-4-performance-tuning-monitoring-and-troublesho
php among other places.

Launching Payara on System Boot

The Dataverse Software installation script will start Payara if necessary, but you may find the following scripts helpful to launch Payara start automatically on boot. They were originally written for Glassfish but have been adjusted for Payara.

- This `Systemd file` may be serve as a reference for systems using Systemd (such as RHEL/derivative or Debian 10, Ubuntu 16+)
- This `init script` may be useful for RHEL/derivative or Ubuntu >= 14 if you're using a Payara service account, or
- This `Payara init script` may be helpful if you're just going to run Payara as root (not recommended).

It is not necessary for Payara to be running before you execute the Dataverse Software installation script; it will start Payara for you.

Please note that you must run Payara in an English locale. If you are using something like `LANG=de_DE.UTF-8`, ingest of tabular data will fail with the message "RoundRoutines:decimal separator no in right place".

Also note that Payara may utilize more than the default number of file descriptors, especially when running batch jobs such as harvesting. We have increased ours by adding ulimit -n 32768 to our Payara init script. On operating systems which use systemd such as RHEL/derivative, file descriptor limits may be increased by adding a line like LimitNOFILE=32768 to the systemd unit file. You may adjust the file descriptor limits on running processes by using the prlimit utility:

```
# sudo prlimit --pid pid --nofile=32768:32768
```

6.3.4 PostgreSQL

PostgreSQL 17 is the version we test against and, with the v6.12 release, 17+ is required for new installations.

Installing PostgreSQL

For example, to install PostgreSQL 17 under RHEL9/derivative:

```
# sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-9-x86_64/
↪pgdg-redhat-repo-latest.noarch.rpm
# sudo dnf check-update
# sudo dnf install -y postgresql17-server
# sudo /usr/pgsql-17/bin/postgresql-17-setup initdb
# sudo /usr/bin/systemctl start postgresql-17
# sudo /usr/bin/systemctl enable postgresql-17
```

You may need to disable the OS's built-in postgresql module with `dnf -qy module disable postgresql`.

Configuring Database Access for the Dataverse Installation (and the Dataverse Software Installer)

- The application and the installer script will be connecting to PostgreSQL over TCP/IP, using password authentication. In this section we explain how to configure PostgreSQL to accept these connections.
- If PostgreSQL is running on the same server as Payara, find the localhost (127.0.0.1) entry that's already in the `pg_hba.conf` and modify it to look like this:

  ```
  host all all 127.0.0.1/32 md5
  ```

  Once you are done with the prerequisites and run the installer script (documented here: *Installation*) it will ask you to enter the address of the Postgres server. Simply accept the default value `127.0.0.1` there.
- The Dataverse Software installer script will need to connect to PostgreSQL **as the admin user**, in order to create and set up the database that the Dataverse installation will be using. If for whatever reason it is failing to connect (for example, if you don't know/remember what your Postgres admin password is), you may choose to temporarily disable all the access restrictions on localhost connections, by changing the above line to:

  ```
  host all all 127.0.0.1/32 trust
  ```

  Note that this rule opens access to the database server **via localhost only**. Still, in a production environment, this may constitute a security risk. So you will likely want to change it back to "md5" once the installer has finished.
- If the Dataverse installation is running on a different server, you will need to add a new entry to the `pg_hba.conf` granting it access by its network address:

  ```
  host all all [ADDRESS]      255.255.255.255 md5
  ```

  Where `[ADDRESS]` is the numeric IP address of the Payara server. Enter this address when the installer asks for the PostgreSQL server address.
- In some distributions, PostgreSQL is pre-configured so that it doesn't accept network connections at all. Check that the `listen_address` line in the configuration file `postgresql.conf` is not commented out and looks like this:

  ```
  listen_addresses='*'
  ```

  The file `postgresql.conf` will be located in the same directory as the `pg_hba.conf` above.
- **Important: PostgreSQL must be restarted** for the configuration changes to take effect! On RHEL9/derivative and similar (provided you installed Postgres as instructed above):

```
# systemctl restart postgresql-17
```

On MacOS X a "Reload Configuration" icon is usually supplied in the PostgreSQL application folder. Or you could look up the process id of the PostgreSQL postmaster process, and send it the SIGHUP signal:

```
kill -1 PROCESS_ID
```

6.3.5 Solr

The Dataverse software search index is powered by Solr.

Supported Versions

The Dataverse software has been tested with Solr version 9.8.0. Future releases in the 9.x series are likely to be compatible. Please get in touch (*Getting Help*) if you are having trouble with a newer version.

Installing Solr

You should not run Solr as root. Create a user called `solr` and a directory to install Solr into:

```
useradd solr
mkdir /usr/local/solr
chown solr:solr /usr/local/solr
```

Become the `solr` user and then download and configure Solr:

```
su - solr
cd /usr/local/solr
wget https://archive.apache.org/dist/solr/solr/9.8.0/solr-9.8.0.tgz
tar xvzf solr-9.8.0.tgz
cd solr-9.8.0
cp -r server/solr/configsets/_default server/solr/collection1
```

You should already have a "dvinstall.zip" file that you downloaded from https://github.com/IQSS/dataverse/releases . Unzip it into `/tmp`. Then copy the files into place:

```
cp /tmp/dvinstall/schema*.xml /usr/local/solr/solr-9.8.0/server/solr/collection1/conf
cp /tmp/dvinstall/solrconfig.xml /usr/local/solr/solr-9.8.0/server/solr/collection1/conf
```

Note: The Dataverse Project team has customized Solr to boost results that come from certain indexed elements inside the Dataverse installation, for example prioritizing results from Dataverse collections over Datasets. If you would like to remove this, edit your `solrconfig.xml` and remove the `<str name="qf">` element and its contents. If you have ideas about how this boosting could be improved, feel free to contact us through our Google Group https://groups.google.com/forum/#!forum/dataverse-dev .

A Dataverse installation requires a change to the `jetty.xml` file that ships with Solr. Edit `/usr/local/solr/solr-9.8.0/server/etc/jetty.xml` , increasing `requestHeaderSize` from `8192` to `102400`

Solr will warn about needing to increase the number of file descriptors and max processes in a production environment but will still run with defaults. We have increased these values to the recommended levels by adding ulimit -n 65000 to the init script, and the following to `/etc/security/limits.conf`:

```
solr soft nproc 65000
solr hard nproc 65000
solr soft nofile 65000
solr hard nofile 65000
```

On operating systems which use systemd such as RHEL/derivative, you may then add a line like LimitNOFILE=65000 for the number of open file descriptors and a line with LimitNPROC=65000 for the max processes to the systemd unit file, or adjust the limits on a running process using the prlimit tool:

```
# sudo prlimit --pid pid --nofile=65000:65000
```

Solr launches asynchronously and attempts to use the `lsof` binary to watch for its own availability. Installation of this package isn't required but will prevent a warning in the log at startup:

```
# dnf install lsof
```

Finally, you need to tell Solr to create the core "collection1" on startup:

```
echo "name=collection1" > /usr/local/solr/solr-9.8.0/server/solr/collection1/core.
↪properties
```

Dataverse collection ("dataverse") page uses Solr very heavily. On a busy instance this may cause the search engine to become the performance bottleneck, making these pages take increasingly longer to load, potentially affecting the overall performance of the application and/or causing Solr itself to crash. If this is observed on your instance, we recommend uncommenting the following lines in the `<circuitBreaker ...>` section of the `solrconfig.xml` file:

```
<str name="memEnabled">true</str>
<str name="memThreshold">75</str>
```

and:

```
<str name="cpuEnabled">true</str>
<str name="cpuThreshold">75</str>
```

This will activate Solr "circuit breaker" mechanisms that make it start dropping incoming requests with the HTTP code 503 when it starts experiencing load issues. As of Dataverse 6.1, the collection page will recognize this condition and display a customizeable message to the users informing them that the search engine is unavailable because of heavy load, with the assumption that the condition is transitive and suggesting that they try again later. This is still an inconvenience to the users, but still a more graceful handling of the problem, rather than letting the pages time out or causing crashes. You may need to experiment and adjust the threshold values defined in the lines above.

If this becomes a common issue, another temporary workaround an admin may choose to use is to enable the following setting:

```
curl -X PUT -d true "http://localhost:8080/api/admin/settings/:DisableSolrFacets"
```

This will make the collection page show the search results without the usual search facets on the left side of the page. Another customizeable message will be shown in that column informing the users that facets are temporarily unavailable. Generating these facets is more resource-intensive for Solr than the main search results themselves, so applying this measure will significantly reduce the load on the search engine.

Solr Init Script

Please choose the right option for your underlying Linux operating system. It will not be necessary to execute both!

For systems running systemd (like RedHat or derivatives since 7, Debian since 9, Ubuntu since 15.04), as root, download `solr.service` and place it in `/tmp`. Then start Solr and configure it to start at boot with the following commands:

```
cp /tmp/solr.service /etc/systemd/system
systemctl daemon-reload
systemctl start solr.service
systemctl enable solr.service
```

For systems using init.d (like CentOS 6), download this `Solr init script` and place it in `/tmp`. Then start Solr and configure it to start at boot with the following commands:

```
cp /tmp/solr /etc/init.d
service start solr
chkconfig solr on
```

Securing Solr

As of version 9.3.0, Solr listens solely on localhost for security reasons. If your installation will run Solr on its own host, you will need to edit `bin/solr.in.sh`, setting `JETTY_HOST` to the external IP address of your Solr server to tell Solr to accept external connections.

We strongly recommend that you also use a firewall to block access to the Solr port (8983) from outside networks. It is **very important** not to allow direct access to the Solr API from outside networks! Otherwise, any host that can reach Solr can add or delete data, search unpublished data, and even reconfigure Solr. For more information, please see https://solr.apache.org/guide/solr/latest/deployment-guide/securing-solr.html

We additionally recommend that the Solr service account's shell be disabled, as it isn't necessary for daily operation:

```
# usermod -s /sbin/nologin solr
```

For Solr upgrades or further configuration you may temporarily re-enable the service account shell:

```
# usermod -s /bin/bash solr
```

or simply prepend each command you would run as the Solr user with "sudo -u solr":

```
# sudo -u solr command
```

Finally, we would like to reiterate that it is simply never a good idea to run Solr as root! Running the process as a non-privileged user would substantially minimize any potential damage even in the event that the instance is compromised.

6.3.6 jq

Installing jq

`jq` is a command line tool for parsing JSON output that is used by the Dataverse Software installation script. It is available in the `appstream` repository:

```
# dnf install epel-release
# dnf install jq
```

or you may install the latest binary for your OS and platform, available from https://github.com/jqlang/jq/releases

6.3.7 ImageMagick

The Dataverse Software uses ImageMagick to generate thumbnail previews of PDF files. This is an optional component, meaning that if you don't have ImageMagick installed, there will be no thumbnails for PDF files, in the search results and on the dataset pages; but everything else will be working. (Thumbnail previews for non-PDF image files are generated using standard Java libraries and do not require any special installation steps).

Installing and configuring ImageMagick

On a Red Hat or derivative Linux distribution, you can install ImageMagick with something like:

```
# dnf install ImageMagick
```

(most RedHat systems will have it pre-installed). When installed using standard `dnf` mechanism, above, the executable for the ImageMagick convert utility will be located at `/usr/bin/convert`. No further configuration steps will then be required.

If the installed location of the convert executable is different from `/usr/bin/convert`, you will also need to specify it in your Payara configuration using the JVM option, below. For example:

```
<jvm-options>-Ddataverse.path.imagemagick.convert=/opt/local/bin/convert</jvm-options>
```

(see the *Configuration* section for more information on the JVM options)

6.3.8 R

The Dataverse Software uses R to handle tabular data files. The instructions below describe a **minimal** R Project installation. It will allow you to ingest R (.RData) files as tabular data and to export tabular data as .RData files. R can be considered an optional component, meaning that if you don't have R installed, you will still be able to run and use the Dataverse Software - but the functionality specific to tabular data mentioned above will not be available to your users.

Installing R

For RHEL/derivative, the EPEL distribution is strongly recommended:

If dnf isn't configured to use EPEL repositories (https://fedoraproject.org/wiki/EPEL):

RHEL9/derivative users can install the epel-release RPM:

```
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
```

RHEL8/derivative users can install the epel-release RPM:

```
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

Rocky 9 users will need te enable the CodeReady-Builder repository:

```
dnf config-manager --enable crb
```

RHEL 8 users will need to enable the CodeReady-Builder repository:

```
subscription-manager repos --enable codeready-builder-for-rhel-8-x86_64-rpms
```

Rocky 8 users will need to enable the PowerTools repository:

```
dnf config-manager --enable powertools
```

To compile the modules below, on a RHEL/Rocky-based system you'll minimally need the following packages:

    dnf install openssl-devel libcurl-devel

Finally, install R with dnf:

```
dnf install R-core R-core-devel
```

Installing the required R libraries

The following R packages (libraries) are required:

```
R2HTML
rjson
DescTools
Rserve
haven
```

Install them following the normal R package installation procedures. For example, with the following R commands:

```
install.packages("R2HTML", repos="https://cloud.r-project.org/", lib="/usr/lib64/R/
↪library" )
install.packages("rjson", repos="https://cloud.r-project.org/", lib="/usr/lib64/R/library
↪" )
install.packages("DescTools", repos="https://cloud.r-project.org/", lib="/usr/lib64/R/
↪library" )
install.packages("Rserve", repos="https://cloud.r-project.org/", lib="/usr/lib64/R/
↪library" )
install.packages("haven", repos="https://cloud.r-project.org/", lib="/usr/lib64/R/library
↪" )
```

Rserve

The Dataverse Software uses Rserve to communicate to R. Rserve is installed as a library package, as described in the step above. It runs as a daemon process on the server, accepting network connections on a dedicated port. This requires some extra configuration and we provide a script for setting it up.

You'll want to obtain local copies of the Rserve setup files found in https://github.com/IQSS/dataverse/tree/master/scripts/r/rserve either by cloning a local copy of the IQSS repository: git clone -b master https://github.com/IQSS/dataverse.git or by downloading the files individually.

Run the script as follows (as root):

```
cd <DATAVERSE SOURCE TREE>/scripts/r/rserve
./rserve-setup.sh
```

The setup script will create a system user rserve that will run the daemon process. It will install the startup script for the daemon (/etc/init.d/rserve), so that it gets started automatically when the system boots. This is an init.d-style startup file. If this is a RedHat/CentOS 7 system, you may want to use the `rserve.service` systemd unit file instead. Copy it into the /usr/lib/systemd/system/ directory, then:

```
# systemctl daemon-reload
# systemctl enable rserve
# systemctl start rserve
```

Note that the setup will also set the Rserve password to "rserve". Rserve daemon runs under a non-privileged user id, so there's not much potential for security damage through unauthorized access. It is however still a good idea **to change the password**. The password is specified in /etc/Rserv.pwd. You can consult Rserve documentation for more information on password encryption and access security.

You should already have the following 4 JVM options added to your domain.xml by the Dataverse installer:

```
<jvm-options>-Ddataverse.rserve.host=localhost</jvm-options>
<jvm-options>-Ddataverse.rserve.port=6311</jvm-options>
<jvm-options>-Ddataverse.rserve.user=rserve</jvm-options>
<jvm-options>-Ddataverse.rserve.password=rserve</jvm-options>
```

If you have changed the password, make sure it is correctly specified in the dataverse.rserve.password option above. If Rserve is running on a host that's different from your Dataverse installation, change the dataverse.rserve.host option above as well (and make sure the port 6311 on the Rserve host is not firewalled from your Dataverse installation host).

6.3.9 Counter Processor

Counter Processor is required to enable Make Data Count metrics in a Dataverse installation. See the *Make Data Count* section of the Admin Guide for a description of this feature. Counter Processor is open source and we will be downloading it from https://github.com/gdcc/counter-processor

Installing Counter Processor

A scripted installation using Ansible is mentioned in the *Make Data Count* section of the Developer Guide.

As root, download and install Counter Processor:

```
cd /usr/local
wget https://github.com/gdcc/counter-processor/archive/refs/tags/v1.06.tar.gz
tar xvfz v1.06.tar.gz
cd /usr/local/counter-processor-1.06
```

Installing GeoLite Country Database

Counter Processor can report per country results if the optional GeoLite Country Database is installed. At present, this database is free but to use it one must signing an agreement (EULA) with MaxMind. (The primary concern appears to be that individuals can opt-out of having their location tracked via IP address and, due to various privacy laws, MaxMind needs a way to comply with that for products it has "sold" (for no cost in this case). Their agreement requires you to either configure automatic updates to the GeoLite Country database or be responsible on your own for managing take down notices.) The process required to sign up, download the database, and to configure automated updating is described at https://blog.maxmind.com/2019/12/18/significant-changes-to-accessing-and-using-geolite2-databases/ and the links from that page.

As root, change to the Counter Processor directory you just created, download the GeoLite2-Country tarball from MaxMind, untar it, and copy the geoip database into place:

```
<download or move the GeoLite2-Country.tar.gz to the /usr/local/counter-processor-1.06␣
↪directory>
tar xvfz GeoLite2-Country.tar.gz
cp GeoLite2-Country_*/GeoLite2-Country.mmdb maxmind_geoip
```

Creating a counter User

As root, create a "counter" user and change ownership of Counter Processor directory to this new user:

```
useradd counter
chown -R counter:counter /usr/local/counter-processor-1.06
```

Installing Counter Processor Python Requirements

Counter Processor version 1.06 requires Python 3.7 or higher.

The following commands are intended to be run as root but we are aware that Pythonistas might prefer fancy virtualenv or similar setups. Pull requests are welcome to improve these steps!

Install Python 3.9:

```
dnf install python3
```

Install Counter Processor Python requirements:

```
python3.9 -m ensurepip
cd /usr/local/counter-processor-1.06
pip3 install -r requirements.txt
```

See the *Make Data Count* section of the Admin Guide for how to configure and run Counter Processor.

6.3.10 Next Steps

Now that you have all the prerequisites in place, you can proceed to the *Installation* section.

6.4 Installation

Now that the *Prerequisites* are in place, we are ready to execute the Dataverse Software installation script (the "installer") and verify that the installation was successful by logging in with a "superuser" account.

Contents:

- *Running the Dataverse Software Installer*
- *Logging In*
  - *Superuser Account*
- *Troubleshooting*
  - *Dataset Cannot Be Published*
  - *Got ERR_ADDRESS_UNREACHABLE While Navigating on Interface or API Calls*
  - *UnknownHostException While Deploying*
- *Fresh Reinstall*
  - *Drop database*
  - *Clear Solr*
  - *Deleting Uploaded Files*
  - *Rerun Installer*
- *Getting Support for Installation Trouble*

6.4.1 Running the Dataverse Software Installer

A scripted, interactive installer is provided. This script will configure your app server environment, create the database, set some required options and start the application. Some configuration tasks will still be required after you run the installer! So make sure to consult the next section.

As mentioned in the *Prerequisites* section, RHEL or a derivative such as RockyLinux or AlmaLinux is recommended. (The installer is also known to work on Mac OS X for setting up a development environment.)

Generally, the installer has a better chance of succeeding if you run it against a freshly installed Payara node that still has all the default configuration settings. In any event, please make sure that it is still configured to accept http connections on port 8080 - because that's where the installer expects to find the application once it's deployed.

You should have already downloaded the installer from https://github.com/IQSS/dataverse/releases when setting up and starting Solr under the *Prerequisites* section. Again, it's a zip file with "dvinstall" in the name.

Unpack the zip file - this will create the directory `dvinstall`.

Important: The installer will need to use the PostgreSQL command line utility `psql` in order to configure the database. If the executable is not in your system PATH, the installer will try to locate it on your system. However, we strongly recommend that you check and make sure it is in the PATH. This is especially important if you have multiple versions of PostgreSQL installed on your system. Make sure the psql that came with the version that you want to use with your Dataverse installation is the first on your path. If you are *running* multiple PostgreSQL servers, make sure you know the port number of the one you want to use, as the installer will need it in order to connect to the database (the first PostgreSQL distribution installed on your system is likely using the default port 5432; but the second will likely be on 5433, etc.) Does every word in this paragraph make sense? If it does, great - because you definitely need to be comfortable with basic system tasks in order to install the Dataverse Software. If not - if you don't know how to check where your PostgreSQL is installed, or what port it is running on, or what a $PATH is… it's not too late to stop. Because it will most likely not work. And if you contact us for help, these will be the questions we'll be asking you - so, again, you need to be able to answer them comfortably for it to work.

It is no longer necessary to run the installer as root!

Just make sure the user running the installer has write permission to:

- /usr/local/payara7/glassfish/lib
- /usr/local/payara7/glassfish/domains/domain1
- the current working directory of the installer (it currently writes its logfile there), and
- your jvm-option specified files.dir

The only reason to run Payara as root would be to allow Payara itself to listen on the default HTTP(S) ports 80 and 443, or any other port below 1024. However, it is simpler and more secure to run Payara run on its default port of 8080 and hide it behind an Apache Proxy, via AJP, running on port 80 or 443. This configuration is required if you're going to use Shibboleth authentication. See more discussion on this here: *Shibboleth*.)

Read the installer script directions like this:

```
$ cd dvinstall
$ less README_python.txt
```

Alternatively you can download `README_python.txt` from this guides.

Follow the instructions in the text file.

The script will prompt you for some configuration values. If this is a test/evaluation installation, it may be possible to accept the default values provided for most of the settings:

- Internet Address of your host: localhost
- Payara Directory: /usr/local/payara7

- Payara User: current user running the installer script
- Administrator email address for this Dataverse installation: (none)
- SMTP (mail) server to relay notification messages: localhost
- Postgres Server Address: [127.0.0.1]
- Postgres Server Port: 5432
- Postgres ADMIN password: secret
- Name of the Postgres Database: dvndb
- Name of the Postgres User: dvnapp
- Postgres user password: secret
- Remote Solr indexing service: LOCAL
- Rserve Server: localhost
- Rserve Server Port: 6311
- Rserve User Name: rserve
- Rserve User Password: rserve
- Administration Email address for the installation;
- Postgres admin password - We'll need it in order to create the database and user for the Dataverse Software installer to use, without having to run the installer as root. If you don't know your Postgres admin password, you may simply set the authorization level for localhost to "trust" in the PostgreSQL `pg_hba.conf` file (See the PostgreSQL section in the Prerequisites). If this is a production environment, you may want to change it back to something more secure, such as "password" or "md5", after the installation is complete.
- Network address of a remote Solr search engine service (if needed) - In most cases, you will be running your Solr server on the same host as the Dataverse Software application (then you will want to leave this set to the default value of `LOCAL`). But in a serious production environment you may set it up on a dedicated separate server.

If desired, these default values can be configured by creating a `default.config` (example `here`) file in the installer's working directory with new values (if this file isn't present, the above defaults will be used).

This allows the installer to be run in non-interactive mode (with `./install.py -y -f > install.out 2> install.err`), which can allow for easier interaction with automated provisioning tools.

All the Payara configuration tasks performed by the installer are isolated in the shell script `dvinstall/as-setup.sh` (as `asadmin` commands).

While Postgres can accomodate usernames and database names containing hyphens, it is strongly recommended to use only alphanumeric characters.

IMPORTANT: As a security measure, the `as-setup.sh` script stores passwords as "aliases" rather than plain-text. If you change your database password, for example, you will need to update the alias with `asadmin update-password-alias dataverse.db.password`, for example. Here is a list of the password aliases that are set by the installation process and entered into Payara's `domain.xml` file:

- `dataverse.db.password`
- `doi_password_alias`
- `rserve_password_alias`

For more information, please see https://docs.payara.fish/documentation/payara-server/password-aliases/password-alias-asadmin-commands.html

IMPORTANT: The installer will also ask for an external site URL for the Dataverse installation. It is *imperative* that this value be supplied accurately, or a long list of functions will be inoperable, including:

- email confirmation links
- password reset links
- generating a Private URL
- exporting to Schema.org format (and showing JSON-LD in HTML's <meta/> tag)
- exporting to DDI format
- which Dataverse installation an "external tool" should return to
- URLs embedded in SWORD API responses

The supplied site URL will be saved under the JVM option *dataverse.siteUrl*.

IMPORTANT: Please note, that "out of the box" the installer will configure the Dataverse installation to leave unrestricted access to the administration APIs from (and only from) localhost. Please consider the security implications of this arrangement (anyone with shell access to the server can potentially mess with your Dataverse installation). An alternative solution would be to block open access to these sensitive API endpoints completely; and to only allow requests supplying a pre-defined "unblock token" (password). If you prefer that as a solution, please consult the supplied script `post-install-api-block.sh` for examples on how to set it up. See also "Securing Your Installation" under the *Configuration* section.

The Dataverse Software uses JHOVE to help identify the file format (CSV, PNG, etc.) for files that users have uploaded. The installer places files called `jhove.conf` and `jhoveConfig.xsd` into the directory `/usr/local/payara7/glassfish/domains/domain1/config` by default and makes adjustments to the jhove.conf file based on the directory into which you chose to install Payara.

6.4.2 Logging In

Out of the box, Payara runs on port 8080 and 8181 rather than 80 and 443, respectively, so visiting http://localhost:8080 (substituting your hostname) should bring up a login page. See the *Shibboleth* page for more on ports, but for now, let's confirm we can log in by using port 8080. Poke a temporary hole in your firewall, if needed.

Superuser Account

We'll use the superuser account created by the installer to make sure you can log into the Dataverse installation. For more on the difference between being a superuser and having the "Admin" role, read about configuring the root Dataverse collection in the *Configuration* section.

(The `dvinstall/setup-all.sh` script, which is called by the installer sets the password for the superuser account account and the username and email address come from a file it references at `dvinstall/data/user-admin.json`.)

Use the following credentials to log in:

- URL: http://localhost:8080
- username: dataverseAdmin
- password: admin

Congratulations! You have a working Dataverse installation. Soon you'll be tweeting at @dataverseorg asking to be added to the map at https://dataverse.org :)

Trouble? See if you find an answer in the troubleshooting section below.

Next you'll want to check out the *Configuration* section, especially the section on security which reminds you to change the password above.

6.4.3 Troubleshooting

If the following doesn't apply, please get in touch as explained in *Getting Help*.

Dataset Cannot Be Published

Check to make sure you used a fully qualified domain name when installing the Dataverse Software. You can change the `dataverse.fqdn` JVM option after the fact per the *Configuration* section.

Got ERR_ADDRESS_UNREACHABLE While Navigating on Interface or API Calls

If you are receiving an `ERR_ADDRESS_UNREACHABLE` while navigating the GUI or making an API call, make sure the `siteUrl` JVM option is defined. For details on how to set `siteUrl`, please refer to *dataverse.siteUrl* from the *Configuration* section. For context on why setting this option is necessary, refer to *dataverse.fqdn* from the *Configuration* section.

UnknownHostException While Deploying

If you are seeing "Caused by: java.net.UnknownHostException: myhost: Name or service not known" in server.log and your hostname is "myhost" the problem is likely that "myhost" doesn't appear in `/etc/hosts`. See also https://stackoverflow.com/questions/21817809/glassfish-exception-during-deployment-project-with-stateful-ejb/21850873#21850873

6.4.4 Fresh Reinstall

Early on when you're installing the Dataverse Software, you may think, "I just want to blow away what I've installed and start over." That's fine. You don't have to uninstall the various components like Payara, PostgreSQL and Solr, but you should be conscious of how to clear out their data. For Payara, a common helpful process is to:

- Stop Payara;
- Remove the `generated`, `lib/databases` and `osgi-cache` directories from the `domain1` directory;
- Start Payara

Drop database

In order to drop the database, you have to stop Payara, which will have open connections. Before you stop Payara, you may as well undeploy the war file. First, find the name like this:

```
./asadmin list-applications
```

Then undeploy it like this:

```
./asadmin undeploy dataverse-VERSION
```

Stop Payara with the init script provided in the *Prerequisites* section or just use:

```
./asadmin stop-domain
```

With Payara down, you should now be able to drop your database and recreate it:

```
psql -U dvnapp -c 'DROP DATABASE "dvndb"' template1
```

Clear Solr

The database is fresh and new but Solr has stale data it in. Clear it out with this command:

```
curl http://localhost:8983/solr/collection1/update/json?commit=true -H "Content-type:
application/json" -X POST -d "{\"delete\":  { \"query\":\"*:*\"}}"
```

Deleting Uploaded Files

The path below will depend on the value for `dataverse.files.directory` as described in the *Configuration* section:

```
rm -rf /usr/local/payara7/glassfish/domains/domain1/files
```

Rerun Installer

With all the data cleared out, you should be ready to rerun the installer per above.

For more on this topic, see *Rebuilding Your Dev Environment* section of the Developer Guide.

6.4.5 Getting Support for Installation Trouble

See *Getting Help*.

6.5 Configuration

Now that you've successfully logged into your Dataverse installation with a superuser account after going through a basic *Installation*, you'll need to secure and configure your installation.

Settings within your Dataverse installation itself are managed via JVM options or by manipulating values in the `setting` table directly or through API calls.

Once you have finished securing and configuring your Dataverse installation, you may proceed to the *Admin Guide* for more information on the ongoing administration of a Dataverse installation. Advanced configuration topics are covered in the *Shibboleth* and *OAuth Login Options* sections.

Contents:

- *Securing Your Installation*
  - *Changing the Superuser Password*
  - *Blocking API Endpoints*
  - *Forcing HTTPS*
  - *Recording User IP Addresses*
  - *Using X-Forwarded-Proto for Signed URLs*
  - *Privacy Considerations*

        - *Email Privacy*
    - *Additional Recommendations*
        - *Run Payara as a User Other Than Root*
        - *Restricting Payara's Ports to localhost*
        - *Secure Password Storage*
        - *Enforce Strong Passwords for User Accounts*
        - *SameSite Cookie Attribute*
    - *Cross-Origin Resource Sharing (CORS)*
    - *Ongoing Security of Your Installation*
        - *New Releases*
        - *Security Advisories*
    - *Reporting Security Issues*
- *Network Ports*
- *Root Dataverse Collection Permissions*
- *Persistent Identifiers and Publishing Datasets*
    - *Testing PID Providers*
    - *Configuring PID Providers*
        - *Global Settings*
        - *dataverse.pid.providers*
        - *dataverse.pid.default-provider*
        - *dataverse.spi.pidproviders.directory*
        - *Per-Provider Settings*
        - *dataverse.pid.*.type*
        - *dataverse.pid.*.label*
        - *dataverse.pid.*.authority*
        - *dataverse.pid.*.shoulder*
        - *dataverse.pid.*.identifier-generation-style*
        - *dataverse.pid.*.datafile-pid-format*
        - *dataverse.pid.*.managed-list*
        - *dataverse.pid.*.excluded-list*
        - *DataCite-specific Settings*
        - *dataverse.pid.*.datacite.mds-api-url*
        - *dataverse.pid.*.datacite.rest-api-url*
        - *dataverse.pid.*.datacite.username*
        - *dataverse.pid.*.datacite.password*

        - *dataverse.feature.only-update-datacite-when-needed*
        - *CrossRef-specific Settings*
        - *dataverse.pid.*.crossref.url*
        - *dataverse.pid.*.crossref.rest-api-url*
        - *dataverse.pid.*.crossref.username*
        - *dataverse.pid.*.crossref.password*
        - *dataverse.pid.*.crossref.depositor*
        - *dataverse.pid.*.crossref.depositor-email*
        - *EZId-specific Settings*
        - *dataverse.pid.*.ezid.api-url*
        - *dataverse.pid.*.ezid.username*
        - *dataverse.pid.*.ezid.password*
        - *PermaLink-specific Settings*
        - *dataverse.pid.*.permalink.base-url*
        - *dataverse.pid.*.permalink.separator*
        - *Handle-specific Settings*
        - *dataverse.pid.*.handlenet.index*
        - *dataverse.pid.*.handlenet.independent-service*
        - *dataverse.pid.*.handlenet.auth-handle*
        - *dataverse.pid.*.handlenet.key*
        - *dataverse.pid.*.handlenet.path*
        - *dataverse.pid.*.handlenet.passphrase*
    - *Backward-compatibility for Single PID Provider Installations*
        - *Configuring Your Dataverse Installation for a Single DOI Provider*
        - *Configuring Your Dataverse Installation for a Single Handle Provider*
        - *Configuring Your Dataverse Installation for a Single PermaLink Provider*
- *Auth Modes: Local vs. Remote vs. Both*
    - *Local Only Auth*
    - *Both Local and Remote Auth*
    - *Remote Only Auth*
- *Bearer Token Authentication*
- *SMTP/Email Configuration*
- *Database Persistence*
    - *Basic Database Settings*
    - *Advanced Database Settings*

        - *Connection Validation*
        - *Connection & Statement Leaks*
        - *Logging & Slow Performance*
    - *Database Configuration Tips*
        - *Database Connection Recovery*
- *File Storage*
    - *Multi-store Basics*
    - *Labels for File Stores*
    - *File Storage*
    - *Swift Storage*
        - *Setting up Compute with Swift*
    - *Amazon S3 Storage (or Compatible)*
        - *First: Set Up Accounts and Access Credentials*
            - *Preparation When Using Amazon's S3 Service*
            - *Preparation When Using Custom S3-Compatible Service*
            - *Manually Set Up Credentials File*
            - *Console Commands to Set Up Access Configuration*
        - *Second: Configure Your Dataverse Installation to use S3 Storage*
            - *S3 Tagging*
            - *Finalizing S3 Configuration*
            - *List of S3 Storage Options*
            - *Credentials via MicroProfile Config*
            - *Reported Working S3-Compatible Storage*
            - *Migrating from Local Storage to S3*
    - *Trusted Remote Storage*
    - *Globus Storage*
    - *Temporary Upload File Storage*
- *Rate Limiting*
- *Branding Your Installation*
    - *Parts of a Dataverse Installation Webpage*
    - *Installation Name/Brand Name*
    - *Header Block*
        - *Navbar*
            - *Custom Navbar Logo*
            - *About URL*

            - *User Guide URL*
            - *Support URL*
            - *Sign Up*
        - *Custom Header*
    - *Content Block*
        - *Custom Homepage*
    - *Footer Block*
        - *Default Footer*
            - *Footer Copyright*
        - *Custom Footer*
    - *Custom Stylesheet*
- *Internationalization*
    - *Adding Multiple Languages to the Dropdown in the Header*
    - *Allowing the Language Used for Dataset Metadata to be Specified*
    - *Configuring the "lang" Directory*
    - *Creating a languages.zip File*
    - *Load the languages.zip file into your Dataverse Installation*
    - *How to Help Translate the Dataverse Software Into Your Language*
    - *Tools for Translators*
- *Web Analytics Code*
    - *Tracking Button Clicks*
    - *Adding Cookie Consent (for GDPR, etc.)*
- *Configuring Licenses*
    - *Setting the Default License*
    - *Adding Licenses*
        - *Adding Creative Common Licenses*
        - *Adding Open Data Commons Licenses*
        - *Adding Software Licenses*
        - *Adding Country-Specific Licenses*
        - *Contributing to the Collection of Standard Licenses Above*
        - *Adding Custom Licenses*
    - *Removing Licenses*
    - *Disabling Custom Dataset Terms*
- *Sorting licenses*
- *BagIt File Handler*

- *BagIt Export*
  - *Duracloud Configuration*
  - *Local Path Configuration*
  - *Google Cloud Configuration*
  - *S3 Configuration*
  - *BagIt Export API Calls*
  - *PostPublication Workflow*
  - *Configuring bag-info.txt*
- *Going Live: Launching Your Production Deployment*
  - *Letting Search Engines Crawl Your Installation*
    - *Ensure robots.txt Is Not Blocking Search Engines*
    - *Creating a Sitemap and Submitting it to Search Engines*
      - *Creating a Sitemap*
      - *Single Sitemap File*
      - *Multiple Sitemap Files (Sitemap Index File)*
      - *Submitting Your Sitemap to Search Engines*
  - *Putting Your Dataverse Installation on the Map at dataverse.org*
  - *Administration of Your Dataverse Installation*
  - *Setting Up Integrations*
- *Comma-Separated Configuration Values*
- *JVM Options*
  - *dataverse.fqdn*
  - *dataverse.siteUrl*
  - *dataverse.files.directory*
  - *dataverse.files.default-dataset-file-count-limit*
  - *dataverse.files.uploads*
  - *dataverse.files.docroot*
  - *dataverse.auth.password-reset-timeout-in-minutes*
  - *dataverse.db.name*
  - *dataverse.db.user*
  - *dataverse.db.password*
  - *dataverse.db.host*
  - *dataverse.db.port*
  - *dataverse.db.parameters*
  - *dataverse.solr.host*

- *dataverse.solr.port*
- *dataverse.solr.core*
- *dataverse.solr.protocol*
- *dataverse.solr.path*
- *dataverse.solr.min-files-to-use-proxy*
- *dataverse.solr.concurrency.max-async-indexes*
- *dataverse.rserve.host*
- *dataverse.rserve.port*
- *dataverse.rserve.user*
- *dataverse.rserve.password*
- *dataverse.rserve.tempdir*
- *dataverse.dropbox.key*
- *dataverse.path.imagemagick.convert*
- *dataverse.dataAccess.thumbnail.image.limit*
- *dataverse.dataAccess.thumbnail.pdf.limit*
- *Legacy Single PID Provider: dataverse.pid.datacite.mds-api-url*
- *Legacy Single PID Provider: dataverse.pid.datacite.rest-api-url*
- *Legacy Single PID Provider: dataverse.pid.datacite.username*
- *Legacy Single PID Provider: dataverse.pid.datacite.password*
- *Legacy Single PID Provider: dataverse.pid.handlenet.key.path*
- *Legacy Single PID Provider: dataverse.pid.handlenet.key.passphrase*
- *Legacy Single PID Provider: dataverse.pid.handlenet.index*
- *Legacy Single PID Provider: dataverse.pid.permalink.base-url*
- *Legacy Single PID Provider: dataverse.pid.ezid.api-url*
- *Legacy Single PID Provider: dataverse.pid.ezid.username*
- *Legacy Single PID Provider: dataverse.pid.ezid.password*
- *dataverse.timerServer*
- *dataverse.lang.directory*
- *dataverse.files.hide-schema-dot-org-download-urls*
- *dataverse.useripaddresssourceheader*
- *dataverse.person-or-org.assume-comma-in-person-name*
- *dataverse.person-or-org.org-phrase-array*
- *dataverse.api.signing-secret*
- *dataverse.api.allow-incomplete-metadata*
- *dataverse.api.blocked.endpoints*

- *dataverse.api.blocked.policy*
- *dataverse.api.blocked.key*
- *dataverse.legacy.api-response-message-style*
- *dataverse.ui.show-validity-label-when-published*
- *dataverse.ui.show-curation-status-to-all*
- *dataverse.signposting.level1-author-limit*
- *dataverse.signposting.level1-item-limit*
- *dataverse.mail.system-email*
- *dataverse.mail.support-email*
- *dataverse.mail.cc-support-on-contact-email*
- *dataverse.mail.debug*
- *dataverse.mail.mta.**
- *dataverse.ui.allow-review-for-incomplete*
- *dataverse.ui.show-validity-filter*
- *dataverse.spi.exporters.directory*
- *dataverse.netcdf.geo-extract-s3-direct-upload*
- *dataverse.storageuse.disable-storageuse-increments*
- *dataverse.auth.oidc.**
- *dataverse.files.guestbook-at-request*
- *dataverse.bagit.sourceorg.name*
- *dataverse.bagit.sourceorg.address*
- *dataverse.bagit.sourceorg.email*
- *dataverse.bagit.archive-on-version-update*
- *dataverse.files.globus-monitoring-server*
- *dataverse.csl.common-styles*
- *dataverse.localcontexts.url*
- *dataverse.localcontexts.api-key*
- *dataverse.search.services.directory*
- *dataverse.search.default-service*
- *dataverse.cors.origin*
- *dataverse.cors.methods*
- *dataverse.cors.headers.allow*
- *dataverse.cors.headers.expose*
- *dataverse.api.mdc.min-delay-ms*
- *dataverse.legacy.schemaorg-in-html-head*

  - *Linked Data Notifications (LDN) Allowed Hosts*
  - *COAR Notify Relationship Announcement Notify Superusers Only*
- *Feature Flags*
  - *dataverse.feature.api-session-auth*
  - *dataverse.feature.api-bearer-auth*
  - *dataverse.feature.api-bearer-auth-provide-missing-claims*
  - *dataverse.feature.api-bearer-auth-handle-tos-acceptance-in-idp*
  - *dataverse.feature.api-bearer-auth-use-builtin-user-on-id-match*
  - *dataverse.feature.api-bearer-auth-use-shib-user-on-id-match*
  - *dataverse.feature.api-bearer-auth-use-oauth-user-on-id-match*
  - *dataverse.feature.unify-api-response-message-style*
  - *dataverse.feature.do-not-assume-default-license*
  - *dataverse.feature.avoid-expensive-solr-join*
  - *dataverse.feature.add-publicobject-solr-field*
  - *dataverse.feature.reduce-solr-deletes*
  - *dataverse.feature.disable-return-to-author-reason*
  - *dataverse.feature.disable-dataset-thumbnail-autoselect*
  - *dataverse.feature.globus-use-experimental-async-framework*
  - *dataverse.feature.index-harvested-metadata-source*
  - *dataverse.feature.enable-version-note*
  - *dataverse.feature.shibboleth-use-wayfinder*
  - *dataverse.feature.shibboleth-use-localhost*
  - *dataverse.feature.add-local-contexts-permission-check*
  - *dataverse.feature.enable-pid-failure-log*
  - *dataverse.feature.role-assignment-history*
  - *dataverse.feature.only-update-datacite-when-needed*
  - *dataverse.feature.require-embargo-reason*
  - *dataverse.feature.croissant-with-local-reviews*
  - *dataverse.feature.allow-locally-fair-data*
- *Application Server Settings*
  - *http.request-timeout-seconds*
  - *http.cookie-same-site-value*
  - *http.cookie-same-site-enabled*
  - *mp.config.profile*
- *Database Settings*

- *:BlockedApiPolicy (Deprecated)*
- *:BlockedApiEndpoints (Deprecated)*
- *:BlockedApiKey (Deprecated)*
- *:BuiltinUsersKey*
- *:SearchApiRequiresToken*
- *:SystemEmail*
- *:HomePageCustomizationFile*
- *:LogoCustomizationFile*
- *:HeaderCustomizationFile*
- *:DisableRootDataverseTheme*
- *:FooterCustomizationFile*
- *:StyleCustomizationFile*
- *:WebAnalyticsCode*
- *:FooterCopyright*
- *Legacy Single PID Provider: :DoiProvider*
- *Legacy Single PID Provider: :Protocol*
- *Legacy Single PID Provider: :Authority*
- *Legacy Single PID Provider: :Shoulder*
- *Legacy Single PID Provider: :IdentifierGenerationStyle*
- *Legacy Single PID Provider: :DataFilePIDFormat*
- *:FilePIDsEnabled*
- *:AllowEnablingFilePIDsPerCollection*
- *Legacy Single PID Provider: :IndependentHandleService*
- *Legacy Single PID Provider: :HandleAuthHandle*
- *:FileValidationOnPublishEnabled*
- *:ApplicationTermsOfUse*
- *:ApplicationPrivacyPolicyUrl*
- *:ApiTermsOfUse*
- *:ExcludeEmailFromExport*
- *:NavbarAboutUrl*
- *:NavbarGuidesUrl*
- *:GuidesBaseUrl*
- *:GuidesVersion*
- *:NavbarSupportUrl*
- *:MetricsUrl*

- *:MaxFileUploadSizeInBytes*
- *:MultipleUploadFilesLimit*
- *:ZipDownloadLimit*
- *:TabularIngestSizeLimit*
- *:HarvestingClientCallRateLimit*
- *:ZipUploadFilesLimit*
- *:SolrHostColonPort*
- *:SolrFullTextIndexing*
- *:SolrMaxFileSizeForFullTextIndexing*
- *:DisableSolrFacets*
- *:DisableSolrFacetsForGuestUsers*
- *:DisableSolrFacetsWithoutJsession*
- *:DisableUncheckedTypesFacet*
- *:SignUpUrl*
- *:LoginSessionTimeout*
- *:DatasetPublishPopupCustomText*
- *:DatasetPublishPopupCustomTextOnAllVersions*
- *:DatasetSubmitForReviewPopupCustomText*
- *:SearchHighlightFragmentSize*
- *:ScrubMigrationData*
- *:MinutesUntilConfirmEmailTokenExpires*
- *:DefaultAuthProvider*
- *:AllowSignUp*
- *:AllowRemoteAuthSignUp*
- *:FileFixityChecksumAlgorithm*
- *:PVMinLength*
- *:PVMaxLength*
- *:PVNumberOfConsecutiveDigitsAllowed*
- *:PVCharacterRules*
- *:PVNumberOfCharacteristics*
- *:PVDictionaries*
- *:PVGoodStrength*
- *:PVCustomPasswordResetAlertMessage*
- *:ShibPassiveLoginEnabled*
- *:ShibAffiliationAttribute*

- *:ShibAttributeCharacterSetConversionEnabled*
- *:ShibAffiliationOrder*
- *:ShibAffiliationSeparator*
- *:ComputeBaseUrl*
- *:CloudEnvironmentName*
- *:PublicInstall*
- *:UploadMethods*
- *:GuestbookResponsesPageDisplayLimit*
- *:CustomDatasetSummaryFields*
- *:AllowApiTokenLookupViaApi*
- *:ProvCollectionEnabled*
- *:MetricsCacheTimeoutMinutes*
- *:MDCLogPath*
- *:DisplayMDCMetrics*
- *:MDCStartDate*
- *:ContactFeedbackMessageSizeLimit*
- *:Languages*
- *:MetadataLanguages*
- *:InheritParentRoleAssignments*
- *:ChronologicalDateFacets*
- *:CustomZipDownloadServiceUrl*
- *:CreateDataFilesMaxErrorsToDisplay*
- *:WorkflowsAdminIpWhitelist*
- *:PrePublishDatasetWorkflowId*
- *:PostPublishDatasetWorkflowId*
- *:PublishDatasetDisclaimerText*
- *:SubmitForReviewDatasetDisclaimerText*
- *:BagItHandlerEnabled*
- *:BagValidatorJobPoolSize*
- *:BagValidatorMaxErrors*
- *:BagValidatorJobWaitInterval*
- *:ArchiverClassName*
- *:ArchiveOnlyIfEarlierVersionsAreArchived*
- *:ArchiverSettings*
- *:BagGeneratorThreads*

- *:DuraCloudHost*
- *:DuraCloudPort*
- *:DuraCloudContext*
- *:BagItLocalPath*
- *:GoogleCloudBucket*
- *:GoogleCloudProject*
- *:S3ArchiverConfig*
- *:InstallationName*
- *:ExportInstallationAsDistributorOnlyWhenNotSet*
- *:AnonymizedFieldTypeNames*
- *:DatasetChecksumValidationSizeLimit*
- *:DataFileChecksumValidationSizeLimit*
- *:SendNotificationOnDatasetCreation*
- *:CVocConf*
- *:ControlledVocabularyCustomJavaScript*
- *:AllowedCurationLabels*
- *:AllowCustomTermsOfUse*
- *:MaxEmbargoDurationInMonths*
- *:MinRetentionDurationInMonths*
- *:DataverseMetadataValidatorScript*
- *:DataverseMetadataPublishValidationFailureMsg*
- *:DataverseMetadataUpdateValidationFailureMsg*
- *:DatasetMetadataValidatorScript*
- *:DatasetMetadataValidationFailureMsg*
- *:ExternalValidationAdminOverride*
- *:FileCategories*
- *:ShowMuteOptions*
- *:AlwaysMuted*
- *:NeverMuted*
- *:LDNMessageHosts*
- *:LDN_TARGET*
- *:LDNAnnounceRequiredFields*
- *:GlobusAppUrl*
- *:GlobusPollingInterval*
- *:GlobusBatchLookupSize*

- *:GlobusSingleFileTransfer*
- *:WebloaderUrl*
- *:CategoryOrder*
- *:OrderByFolder*
- *:AllowUserManagementOfOrder*
- *:UseStorageQuotas*
- *:StoreIngestedTabularFilesWithVarHeaders*
- *:RateLimitingDefaultCapacityTiers*
- *:RateLimitingCapacityByTierAndAction*
- *:GetExternalSearchUrl*
- *:GetExternalSearchName*
- *:PostExternalSearchUrl*
- *:PostExternalSearchName*

6.5.1 Securing Your Installation

Changing the Superuser Password

The default password for the "dataverseAdmin" superuser account is "admin", as mentioned in the *Installation* section, and you should change it, of course.

Blocking API Endpoints

The *Native API* contains a useful but potentially dangerous set of API endpoints called "admin" that allows you to change system settings, make ordinary users into superusers, and more. The "builtin-users" endpoints let admins do tasks such as creating a local/builtin user account if they know the key defined in *:BuiltinUsersKey*.

By default in the code, most of these API endpoints can be operated on remotely and a number of endpoints do not require authentication. However, the endpoints "admin" and "builtin-users" are limited to localhost out of the box by the installer, using the JvmSettings *dataverse.api.blocked.endpoints* and *dataverse.api.blocked.policy*.

Note

The database settings *:BlockedApiEndpoints (Deprecated)* and *:BlockedApiPolicy (Deprecated)* are deprecated and will be removed in a future version. Please use the JvmSettings mentioned above instead.

It is **very important** to keep the block in place for the "admin" endpoint, and to leave the "builtin-users" endpoint blocked unless you need to access it remotely. Documentation for the "admin" endpoint is spread across the *Native API* section of the API Guide and the *Admin Guide*.

Given how important it is to avoid exposing the "admin" and "builtin-user" APIs, sites using a proxy, e.g. Apache or Nginx, should also consider blocking them through rules in the proxy. The following examples may be useful:

Apache/Httpd Rule:

Rewrite lines added to /etc/httpd/conf.d/ssl.conf. They can be the first lines inserted after the RewriteEngine On statement:

```
RewriteRule   ^/api/(admin|builtin-users)           - [R=403,L]
RewriteRule   ^/api/(v[0-9]*)/(admin|builtin-users) - [R=403,L]
```

Nginx Configuration Rule:

```
location ~ ^/api/(admin|v1/admin|builtin-users|v1/builtin-users) {
    deny all;
    return 403;
}
```

If you are using a load balancer or a reverse proxy, there are some additional considerations. If no additional configurations are made and the upstream is configured to redirect to localhost, the API will be accessible from the outside, as your installation will register as origin the localhost for any requests to the endpoints "admin" and "builtin-users". To prevent this, you have two options:

- If your upstream is configured to redirect to localhost, you will need to set the *JVM option* to one of the following values `%client.name% %datetime% %request% %status% %response.length% %header.referer% %header.x-forwarded-for%` and configure from the load balancer side the chosen header to populate with the client IP address.
- Another solution is to set the upstream to the client IP address. In this case no further configuration is needed.

For more information on configuring blocked API endpoints, see *dataverse.api.blocked.endpoints* and *dataverse.api.blocked.policy* in the JvmSettings documentation.

Note

It's also possible to prevent file uploads via API by adjusting the *:UploadMethods* database setting.

Forcing HTTPS

To avoid having your users send credentials in the clear, it's strongly recommended to force all web traffic to go through HTTPS (port 443) rather than HTTP (port 80). The ease with which one can install a valid SSL cert into Apache compared with the same operation in Payara might be a compelling enough reason to front Payara with Apache. In addition, Apache can be configured to rewrite HTTP to HTTPS with rules such as those found at https://wiki.apache.org/httpd/RewriteHTTPToHTTPS or in the section on *Shibboleth*.

Recording User IP Addresses

By default, the Dataverse installation captures the IP address from which requests originate. This is used for multiple purposes including controlling access to the admin API, IP-based user groups and Make Data Count reporting. When the Dataverse installation is configured behind a proxy such as a load balancer, this default setup may not capture the correct IP address. In this case all the incoming requests will be logged in the access logs, MDC logs etc., as if they are all coming from the IP address(es) of the load balancer itself. Proxies usually save the original address in an added HTTP header, from which it can be extracted. For example, AWS LB records the "true" original address in the standard `X-Forwarded-For` header. If your Dataverse installation is running behind an IP-masking proxy, but you would like to use IP groups, or record the true geographical location of the incoming requests with Make Data Count, you may enable the IP address lookup from the proxy header using the JVM option `dataverse.useripaddresssourceheader`, described further below.

Before doing so however, you must absolutely **consider the security risks involved**! This option must be enabled **only** on a Dataverse installation that is in fact fully behind a proxy that properly, and consistently, adds the `X-Forwarded-For` (or a similar) header to every request it forwards. Consider the implications of activating this option on a Dataverse installation that is not running behind a proxy, *or running behind one, but still accessible from the insecure locations bypassing the proxy*: Anyone can now add the header above to an incoming request, supplying an arbitrary IP address that the Dataverse installation will trust as the true origin of the call. Thus giving an attacker an easy way to, for example, get in a privileged IP group. The implications could be even more severe if an attacker were able to pretend to be coming from `localhost`, if a Dataverse installation is configured to trust localhost connections for unrestricted access to the admin API! We have addressed this by making it so that Dataverse installation should never accept `localhost`, `127.0.0.1`, `0:0:0:0:0:0:0:1` etc. when supplied in such a header. But if you have reasons to still find this risk unacceptable, you may want to consider turning open localhost access to the API off (See *Securing Your Installation* for more information.)

This is how to verify that your proxy or load balancer, etc. is handling the originating address headers properly and securely: Make sure access logging is enabled in your application server (Payara) configuration. (`<http-service access-logging-enabled="true">` in the `domain.xml`). Add the address header to the access log format. For example, on a system behind AWS ELB, you may want to use something like `%client.name% %datetime% %request% %status% %response.length% %header.referer% %header.x-forwarded-for%`. Once enabled, access the Dataverse installation from outside the LB. You should now see the real IP address of your remote client in the access log. For example, something like: `"1.2.3.4" "01/Jun/2020:12:00:00 -0500" "GET /dataverse.xhtml HTTP/1.1" 200 81082 "NULL-REFERER" "128.64.32.16"`

In this example, `128.64.32.16` is your remote address (that you should verify), and `1.2.3.4` is the address of your LB. If you're not seeing your remote address in the log, do not activate the JVM option! Also, verify that all the entries in the log have this header populated. The only entries in the access log that you should be seeing without this header (logged as `"NULL-HEADER-X-FORWARDED-FOR"`) are local requests, made from localhost, etc. In this case, since the request is not coming through the proxy, the local IP address should be logged as the primary one (as the first value in the log entry, `%client.name%`). If you see any requests coming in from remote, insecure subnets without this header - do not use the JVM option!

Once you are ready, enable the *JVM option*. Verify that the remote locations are properly tracked in your MDC metrics, and/or your IP groups are working. As a final test, if your Dataverse installation is allowing unrestricted localhost access to the admin API, imitate an attack in which a malicious request is pretending to be coming from `127.0.0.1`. Try the following from a remote, insecure location:

```
curl https://your.dataverse.edu/api/admin/settings --header "X-FORWARDED-FOR: 127.0.0.1"
```

First of all, confirm that access is denied! If you are in fact able to access the settings api from a location outside the proxy, **something is seriously wrong**, so please let us know, and stop using the JVM option. Otherwise check the access log entry for the header value. What you should see is something like `"127.0.0.1, 128.64.32.16"`. Where the second address should be the real IP of your remote client. The fact that the "fake" `127.0.0.1` you sent over is present in the header is perfectly ok. This is the proper proxy behavior - it preserves any incoming values in the `X-Forwarded-Header`, if supplied, and adds the detected incoming address to it, *on the right*. It is only this rightmost comma-separated value that Dataverse installation should ever be using.

Still feel like activating this option in your configuration? - Have fun and be safe!

Using X-Forwarded-Proto for Signed URLs

If you use a proxy such as Apache or Nginx, or have a firewall such as Anubis, and they are configured to forward traffic to Dataverse over HTTP (i.e. your proxy receives user calls over HTTPS but forwards locally to Dataverse over HTTP), signed URLs, used by external tools and upload apps (such as DVWebloader), are likely to fail unless you configure your proxy to send an X-Forwarded-Proto HTTP Header. This allows Dataverse to recognize that the communication from the user was over HTTPS and that validation of signed URLs should assume they started with https:// (rather than http:// as received from the proxy).

Privacy Considerations

Email Privacy

Out of the box, your Dataverse installation will list email addresses of the contacts for datasets when users visit a dataset page and click the "Export Metadata" button. Additionally, out of the box, the Dataverse installation will list email addresses of Dataverse collection contacts via API (see *View a Dataverse Collection* in the *Native API* section of the API Guide). If you would like to exclude these email addresses from export, set *:ExcludeEmailFromExport* to true.

Additional Recommendations

Run Payara as a User Other Than Root

See the *Payara* section of *Prerequisites* for details and init scripts for running Payara as non-root.

Related to this is that you should remove `/root/.payara/pass` to ensure that Payara isn't ever accidentally started as root. Without the password, Payara won't be able to start as root, which is a good thing.

Restricting Payara's Ports to localhost

In the recommended setup of Dataverse, you do not expose Payara's ports directly to the Internet. Rather, you front Payara with a proxy such as Apache.

If you are running Payara and your proxy on the same server, we recommend having Payara listen only to localhost, which is how your proxy talks to it, with the following command:

```
./asadmin set server-config.network-config.network-listeners.network-listener.
http-listener-1.address=127.0.0.1
```

(You should **NOT** use the configuration option above if you are running in a load-balanced environment, or otherwise have your proxy on a different host than Payara.)

To test that Payara is now only listening on localhost, try hitting port 8080 from the Internet. Payara should not respond.

See also *Network Ports*.

Secure Password Storage

In development or demo scenarios, we suggest not to store passwords in files permanently. We recommend the use of at least environment variables or production-grade mechanisms to supply passwords.

In a production setup, permanently storing passwords as plaintext should be avoided at all cost. Environment variables are dangerous in shared environments and containers, as they may be easily exploited; we suggest not to use them. Depending on your deployment model and environment, you can make use of the following techniques to securely store and access passwords.

Password Aliases

A password alias allows you to have a plaintext reference to an encrypted password stored on the server, with the alias being used wherever the password is needed. This method is especially useful in a classic deployment, as it does not require any external secrets management.

Password aliases are consumable as a MicroProfile Config source and can be referrenced by their name in a property expression. You may also reference them within a variable substitution, e.g. in your `domain.xml`.

Creation example for an alias named *my.alias.name*:

```
echo "AS_ADMIN_ALIASPASSWORD=changeme" > /tmp/p.txt
asadmin create-password-alias --passwordfile "/tmp/p.txt" "my.alias.name"
rm /tmp/p.txt
```

Note: omitting the `--passwordfile` parameter allows creating the alias in an interactive fashion with a prompt.

Secrets Files

Payara has a builtin MicroProfile Config source to consume values from files in a directory on your filesystem. This directory config source is most useful and secure with external secrets management in place, temporarily mounting cleartext passwords as files. Examples are Kubernetes / OpenShift Secrets or tools like Vault Agent.

Please follow the directory config source documentation to learn about its usage.

Cloud Providers

Running Dataverse on a cloud platform or running an external secret management system like Vault enables accessing secrets without any intermediate storage of cleartext. Obviously this is the most secure option for any deployment model, but it may require more resources to set up and maintain - your mileage may vary.

Take a look at cloud sources shipped with Payara to learn about their usage.

Enforce Strong Passwords for User Accounts

Your Dataverse installation only stores passwords (as salted hash, and using a strong hashing algorithm) for "builtin" users. You can increase the password complexity rules to meet your security needs. If you have configured your Dataverse installation to allow login from remote authentication providers such as Shibboleth, ORCID, GitHub or Google, you do not have any control over those remote providers' password complexity rules. See the *Auth Modes: Local vs. Remote vs. Both* section below for more on login options.

Even if you are satisfied with the out-of-the-box password complexity rules the Dataverse Software ships with, for the "dataverseAdmin" account you should use a strong password so the hash cannot easily be cracked through dictionary attacks.

Password complexity rules for "builtin" accounts can be adjusted with a variety of settings documented below. Here's a list:

- *:PVMinLength*

- *:PVMaxLength*
- *:PVNumberOfConsecutiveDigitsAllowed*
- *:PVCharacterRules*
- *:PVNumberOfCharacteristics*
- *:PVDictionaries*
- *:PVGoodStrength*
- *:PVCustomPasswordResetAlertMessage*

SameSite Cookie Attribute

The SameSite cookie attribute is defined in an upcoming revision to RFC 6265 (HTTP State Management Mechanism) called 6265bis ("bis" meaning "repeated"). The possible values are "None", "Lax", and "Strict". "Strict" is intended to help prevent Cross-Site Request Forgery (CSRF) attacks, as described in the RFC proposal and an OWASP cheetsheet. We don't recommend "None" for security reasons.

By default, Payara doesn't send the SameSite cookie attribute, which browsers should interpret as "Lax" according to MDN. Dataverse installations are explicity set to "Lax" out of the box by the installer (in the case of a "classic" installation) or through the base image (in the case of a Docker installation). For classic, see *http.cookie-same-site-value* and *http.cookie-same-site-enabled* for how to change the values. For Docker, you must rebuild the *base image*. See also Payara's documentation for the settings above.

To inspect cookie attributes like SameSite, you can use `curl -s -I http://localhost:8080 | grep JSESSIONID`, for example, looking for the "Set-Cookie" header.

Cross-Origin Resource Sharing (CORS)

For any Dataverse installation using or planning to use advanced features like *big data support* or *file previewers*, CORS must be configured.

To understand what CORS is all about and how it works, the following are recommended reads:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS
- https://corsfix.com/cors-headers
- https://www.caduh.com/blog/understanding-cors
- https://medium.com/@roelljr/demystifying-cors-its-just-http-headers-i-promise-4a02caf460fa

To learn how to configure the Dataverse application to send CORS headers to browsers, these JVM options are relevant:

- *dataverse.cors.origin*
- *dataverse.cors.methods*
- *dataverse.cors.headers.allow*
- *dataverse.cors.headers.expose*

Dataverse will only emit the necessary `Access-Control-*` headers to browsers when CORS has been explicitly enabled via the JVM option *dataverse.cors.origin*.

For any resources to be integrated with Dataverse, find documentation how to set up CORS rules on their end at:

- *Big Data: CORS for S3 buckets*
- GDCC/dataverse-previewers

Ongoing Security of Your Installation

New Releases

Like any application, you should keep up-to-date with new releases of Dataverse, which often contain security fixes.

Dataverse releases are announced on the dataverse-community mailing list, Zulip, and the Dataverse blog.

These announcements link to release notes, which can be found on GitHub:

- Backend: https://github.com/IQSS/dataverse/releases
- Frontend: https://github.com/IQSS/dataverse-frontend/releases

If the release contains a security-related fix, the release notes will mention generically that a fix is included. Some details are provided via private security advisories, described below.

Security Advisories

We highly recommend signing up to receive security advisories from the Dataverse team. These advisories are not archived publicly because we don't want attackers to use exploits against you. For this reason we also control who is subscribed to the security advisory mailing list.

To subscribe, email support@dataverse.org and ask to be added to the Dataverse security advisory mailing list.

Below you'll see a step called *Putting Your Dataverse Installation on the Map at dataverse.org* where you can provide a email address we can use to contact you. We typically automatically add this address to the security list but please feel free to reach out early.

For additional details about security practices by the Dataverse team, see the *Security* section of the Developer Guide.

Reporting Security Issues

If you have a security issue to report, please email it to security@dataverse.org. We usually give credit to security researchers in our release notes. We expect security researchers to look for this information in SECURITY.md (`plain-text`, HTML) and include it there.

6.5.2 Network Ports

Remember how under "Decisions to Make" in the *Preparation* section we mentioned you'll need to make a decision about whether or not to introduce a proxy in front of the Dataverse Software such as Apache or nginx? The time has come to make that decision.

The need to redirect port HTTP (port 80) to HTTPS (port 443) for security has already been mentioned above and the fact that Payara puts these services on 8080 and 8181, respectively, was touched on in the *Installation* section. In production, you don't want to tell your users to use your Dataverse installation on ports 8080 and 8181. You should have them use the standard HTTPS port, which is 443.

Your decision to proxy or not should primarily be driven by which features of the Dataverse Software you'd like to use. If you'd like to use Shibboleth, the decision is easy because proxying or "fronting" Payara with Apache is required. The details are covered in the *Shibboleth* section.

Even if you have no interest in Shibboleth, you may want to front your Dataverse installation with Apache or nginx to simply the process of installing SSL certificates. There are many tutorials on the Internet for adding certs to Apache, including a some notes used by the Dataverse Project team, but the process of adding a certificate to Payara is arduous and not for the faint of heart. The Dataverse Project team cannot provide much help with adding certificates to Payara beyond linking to tips on the web.

Still not convinced you should put Payara behind another web server? Even if you manage to get your SSL certificate into Payara, how are you going to run Payara on low ports such as 80 and 443? Are you going to run Payara as root? Bad idea. This is a security risk. Under "Additional Recommendations" under "Securing Your Installation" above you are advised to configure Payara to run as a user other than root.

There's also the issue of serving a production-ready version of robots.txt. By using a proxy such as Apache, this is a one-time "set it and forget it" step as explained below in the "Going Live" section.

If you are convinced you'd like to try fronting Payara with Apache, the *Shibboleth* section should be good resource for you.

If you really don't want to front Payara with any proxy (not recommended), you can configure Payara to run HTTPS on port 443 like this:

```
./asadmin set server-config.network-config.network-listeners.network-listener.
http-listener-2.port=443
```

What about port 80? Even if you don't front your Dataverse installation with Apache, you may want to let Apache run on port 80 just to rewrite HTTP to HTTPS as described above. You can use a similar command as above to change the HTTP port that Payara uses from 8080 to 80 (substitute `http-listener-1.port=80`). Payara can be used to enforce HTTPS on its own without Apache, but configuring this is an exercise for the reader. Answers here may be helpful: https://stackoverflow.com/questions/25122025/glassfish-v4-java-7-port-unification-error-not-able-to-redirect-http-to

If you are running an installation with Apache and Payara on the same server, and would like to restrict Payara from responding to any requests to port 8080 from external hosts (in other words, not through Apache), you can restrict the AJP listener to localhost only with:

```
./asadmin set server-config.network-config.network-listeners.network-listener.
http-listener-1.address=127.0.0.1
```

You should **NOT** use the configuration option above if you are running in a load-balanced environment, or otherwise have the web server on a different host than the application server.

This security tip is also mentioned at *Restricting Payara's Ports to localhost*.

6.5.3 Root Dataverse Collection Permissions

The user who creates a Dataverse collection is given the "Admin" role on that Dataverse collection. The root Dataverse collection is created automatically for you by the installer and the "Admin" is the superuser account ("dataverseAdmin") we used in the *Installation* section to confirm that we can log in. These next steps of configuring the root Dataverse collection require the "Admin" role on the root Dataverse collection, but not the much more powerful superuser attribute. In short, users with the "Admin" role are subject to the permission system. A superuser, on the other hand, completely bypasses the permission system. You can give non-superusers the "Admin" role on the root Dataverse collection if you'd like them to configure the root Dataverse collection.

In order for non-superusers to start creating Dataverse collections or datasets, you need click "Edit" then "Permissions" and make choices about which users can add Dataverse collections or datasets within the root Dataverse collection. (There is an API endpoint for this operation as well.) Again, the user who creates a Dataverse collection will be granted the "Admin" role on that Dataverse collection. Non-superusers who are not "Admin" on the root Dataverse collection will not be able to do anything useful until the root Dataverse collection has been published.

As the person installing the Dataverse Software, you may or may not be a local metadata expert. You may want to have others sign up for accounts and grant them the "Admin" role at the root Dataverse collection to configure metadata fields, templates, browse/search facets, guestbooks, etc. For more on these topics, consult the *Dataverse Collection Management* section of the User Guide.

6.5.4 Persistent Identifiers and Publishing Datasets

Persistent identifiers (PIDs) are a required and integral part of the Dataverse Software. They provide a URL that is guaranteed to resolve to the datasets or files they represent. The Dataverse Software currently supports creating identifiers using any of several PID types. The most appropriate PIDs for public data are DOIs (e.g., provided by DataCite or EZID) and Handles. Dataverse also supports PermaLinks which could be useful for intranet or catalog use cases. A DOI provider called "FAKE" is recommended only for testing and development purposes.

Dataverse can be configured with one or more PID providers, each of which can mint and manage PIDs with a given protocol (e.g., doi, handle, permalink) using a specific service provider/account (e.g. with DataCite, EZId, or HandleNet) to manage an authority/shoulder combination, aka a "prefix" (PermaLinks also support custom separator characters as part of the prefix), along with an optional list of individual PIDs (with different authority/shoulders) than can be managed with that account.

Dataverse automatically manages assigning PIDs and making them findable when datasets are published. There are also *API calls that allow updating the PID target URLs and metadata of already-published datasets manually if needed*, e.g. if a Dataverse instance is moved to a new URL or when the software is updated to generate additional metadata or address schema changes at the PID service.

Note that while some forms of PIDs (Handles, PermaLinks) are technically case sensitive, common practice is to avoid creating PIDs that differ only by case. Dataverse treats PIDs of all types as case-insensitive (as DOIs are by definition). This means that Dataverse will find datasets (in search, to display dataset pages, etc.) when the PIDs entered do not match the case of the original but will have a problem if two PIDs that differ only by case exist in one instance.

Testing PID Providers

By default, the installer configures the Fake DOI provider as the registration provider. Unlike other DOI Providers, the Fake Provider does not involve any external resolution service and is not appropriate for use beyond development and testing. You may wish instead to test with PermaLinks or with a DataCite test account (which uses DataCite's test infrastructure and will help assure your Dataverse instance can make network connections to DataCite. DataCite requires that you register for a test account, which will have a username, password and your own prefix (please contact support@datacite.org for a test account. You may wish to contact the GDCC instead - GDCC is able to provide DataCite accounts with a group discount and can also provide test accounts.).

Once you receive the login name, password, and prefix for the account, configure the credentials as described below.

Alternately, you may wish to configure other providers for testing:

- EZID is available to University of California scholars and researchers. Testing can be done using the authority 10.5072 and shoulder FK2 with the "apitest" account (contact EZID for credentials) or an institutional account. Configuration in Dataverse is then analogous to using DataCite.
- The PermaLink provider, like the FAKE DOI provider, does not involve an external account. Unlike the Fake DOI provider, the PermaLink provider creates PIDs that begin with "perma:", making it clearer that they are not DOIs, and that do resolve to the local dataset/file page in Dataverse, making them useful for some production use cases. See *Configuring Your Dataverse Installation for a Single PermaLink Provider* and (for the FAKE DOI provider) the *Development Environment* section of the Developer Guide.

Provider-specific configuration is described below.

Once all is configured, you will be able to publish datasets and files, but **the persistent identifiers will not be citable** as they, with the exception of PermaLinks, will not redirect to your dataset page in Dataverse.

Note that any datasets or files created using a test configuration cannot be directly migrated to a production PID provider and would need to be created again once a valid PID Provider(s) are configured.

One you are done testing, to properly configure persistent identifiers for a production installation, an account and associated namespace (e.g. authority/shoulder) must be acquired for a fee from a DOI or HDL provider. (As noted above,

PermaLinks May be appropriate for intranet and catalog uses cases.) **DataCite** (https://www.datacite.org) is the recommended DOI provider (see https://dataversecommunity.global for more on joining DataCite through the Global Dataverse Community Consortium) but **EZID** (http://ezid.cdlib.org) is an option for the University of California according to https://www.cdlib.org/cdlinfo/2017/08/04/ezid-doi-service-is-evolving/ . **Handle.Net** (https://www.handle.net) is the HDL provider.

Once you have your DOI or Handle account credentials and a prefix, configure your Dataverse installation using the settings below.

Configuring PID Providers

There are two required global settings to configure PID providers - the list of ids of providers and which one of those should be the default. Per-provider settings are also required - some that are common to all types and some type specific. All of these settings are defined to be compatible with the MicroProfile specification which means that

1. Any of these settings can be set via system properties (see *JVM Options* for how to do this), environment variables, or other MicroProfile Config mechanisms supported by the app server. See Payara docs for supported sources.
2. Remember to protect your secrets. For passwords, use an environment variable (bare minimum), a password alias named the same as the key (OK) or use the "dir config source" of Payara (best).

   Alias creation example:

   ```
   echo "AS_ADMIN_ALIASPASSWORD=changeme" > /tmp/p.txt
   asadmin create-password-alias --passwordfile /tmp/p.txt dataverse.pid.datacite1.
   ↪datacite.password
   rm /tmp/p.txt
   ```

3. Environment variables follow the key, replacing any dot, colon, dash, etc. into an underscore "_" and all uppercase letters. Example: `dataverse.pid.default-provider` -> `DATAVERSE_PID_DEFAULT_PROVIDER`

Global Settings

The following two global settings are required to configure PID Providers in the Dataverse software:

dataverse.pid.providers

A comma-separated list of the ids of the PID providers to use. IDs should be simple unique text strings, e.g. datacite1, perma1, etc. IDs are used to scope the provider-specific settings but are not directly visible to users.

dataverse.pid.default-provider

The ID of the default PID provider to use.

dataverse.spi.pidproviders.directory

The path to the directory where JAR files containing additional types of PID Providers can be added. Dataverse includes providers that support DOIs (DataCite, EZId, or FAKE), Handles, and PermaLinks. PID provider jar files added to this directory can replace any of these or add new PID Providers.

Per-Provider Settings

Each Provider listed by id in the dataverse.pid.providers setting must be configured with the following common settings and any settings that are specific to the provider type.

dataverse.pid.*.type

The Provider type, currently one of `datacite`, `ezid`, `FAKE`, `hdl`, or `perma`. The type defines which protocol a service supports (DOI, Handle, or PermaLink) and, for DOI Providers, which DOI service is used.

dataverse.pid.*.label

A human-readable label for the provider

dataverse.pid.*.authority

dataverse.pid.*.shoulder

In general, PIDs are of the form `<protocol>:<authority>/<shoulder>*` where * is the portion unique to an individual PID. PID Providers must define the authority and shoulder (with the protocol defined by the `dataverse.pid.*.type` setting) that defines the set of existing PIDs they can manage and the prefix they can use when minting new PIDs. (Often an account with a PID service provider will be limited to using a single authority/shoulder. If your PID service provider account allows more than one combination that you wish to use in Dataverse, configure multiple PID Provider, one for each combination.)

dataverse.pid.*.identifier-generation-style

By default, Pid Providers in Dataverse generate a random 6 character string, pre-pended by the Shoulder if set, to use as the identifier for a Dataset. Set this to `storedProcGenerated` to generate instead a custom *unique* identifier (again pre-pended by the Shoulder if set) through a database stored procedure or function (the assumed default setting is `randomString`). When using the `storedProcGenerated` setting, a stored procedure or function must be created in the database.

As a first example, the script below (downloadable `here`) produces sequential numerical values. You may need to make some changes to suit your system setup, see the comments for more information:

```
-- A script for creating a numeric identifier sequence, and an external
-- stored procedure, for accessing the sequence from inside the application,
-- in a non-hacky, JPA way.

-- NOTE:
```

(continues on next page)

(continued from previous page)

```
-- 1. The database user name "dvnapp" is hard-coded here - it may
-- need to be changed to match your database user name;

-- 2. In the code below, the sequence starts with 1, but it can be adjusted by
-- changing the MINVALUE as needed.

CREATE SEQUENCE datasetidentifier_seq
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 9223372036854775807
  START 1
CACHE 1;

ALTER TABLE datasetidentifier_seq OWNER TO "dvnapp";

-- And now create a PostgreSQL FUNCTION, for JPA to
-- access as a NamedStoredProcedure:

CREATE OR REPLACE FUNCTION generateIdentifierFromStoredProcedure()
RETURNS varchar AS $$
DECLARE
  identifier varchar;
BEGIN
  identifier := nextval('datasetidentifier_seq')::varchar;
  RETURN identifier;
END;
$$ LANGUAGE plpgsql IMMUTABLE;
```

As a second example, the script below (downloadable here) produces sequential 8 character identifiers from a base36 representation of current timestamp.

```
-- A script for creating, through a database stored procedure, sequential
-- 8 character identifiers from a base36 representation of current timestamp.

CREATE OR REPLACE FUNCTION base36_encode(
  IN digits bigint, IN min_width int = 0)
RETURNS varchar AS $$
DECLARE
    chars char[];
    ret varchar;
    val bigint;
BEGIN
    chars := ARRAY[
      '0','1','2','3','4','5','6','7','8','9',
      'a','b','c','d','e','f','g','h','i','j',
      'k','l','m','n','o','p','q','r','s','t',
      'u','v','w','x','y','z'];
    val := digits;
    ret := '';
    IF val < 0 THEN
        val := val * -1;
    END IF;
```

(continues on next page)

(continued from previous page)

```
    WHILE val != 0 LOOP
        ret := chars[(val % 36)+1] || ret;
        val := val / 36;
    END LOOP;

    IF min_width > 0 AND char_length(ret) < min_width THEN
        ret := lpad(ret, min_width, '0');
    END IF;

    RETURN ret;
END;
$$ LANGUAGE plpgsql IMMUTABLE;


CREATE OR REPLACE FUNCTION generateIdentifierFromStoredProcedure()
RETURNS varchar AS $$
DECLARE
    curr_time_msec bigint;
    identifier varchar;
BEGIN
    curr_time_msec := extract(epoch from now())*1000;
    identifier := base36_encode(curr_time_msec);
    RETURN identifier;
END;
$$ LANGUAGE plpgsql IMMUTABLE;
```

Note that the SQL in these examples scripts is Postgres-specific. If necessary, it can be reimplemented in any other SQL flavor - the standard JPA code in the application simply expects the database to have a saved function ("stored procedure") named `generateIdentifierFromStoredProcedure()` returning a single `varchar` argument.

Please note that this setting interacts with the `dataverse.pid.*.datafile-pid-format` setting below to determine how datafile identifiers are generated.

dataverse.pid.*.datafile-pid-format

This setting controls the way that the "identifier" component of a file's persistent identifier (PID) relates to the PID of its "parent" dataset - for a give PID Provider.

By default the identifier for a file is dependent on its parent dataset. For example, if the identifier of a dataset is "TJCLKP", the identifier for a file within that dataset will consist of the parent dataset's identifier followed by a slash ("/"), followed by a random 6 character string, yielding "TJCLKP/MLGWJO". Identifiers in this format are what you should expect if you leave `dataverse.pid.*.datafile-pid-format` undefined or set it to `DEPENDENT` and have not changed the `dataverse.pid.*.identifier-generation-style` setting from its default.

Alternatively, the identifier for File PIDs can be configured to be independent of Dataset PIDs using the setting `INDEPENDENT`. In this case, file PIDs will not contain the PIDs of their parent datasets, and their PIDs will be generated the exact same way that datasets' PIDs are, based on the `dataverse.pid.*.identifier-generation-style` setting described above (random 6 character strings or custom unique identifiers through a stored procedure, pre-pended by any shoulder).

The chart below shows examples from each possible combination of parameters from the two settings. `dataverse.pid.*.identifier-generation-style` can be either `randomString` (the default) or `storedProcGenerated` and `dataverse.pid.*.datafile-pid-format` can be either `DEPENDENT` (the default) or `INDEPENDENT`. In the examples below the "identifier" for the dataset is "TJCLKP" for `randomString` and "100001" for `storedProcGenerated`

(when using sequential numerical values, as described in *dataverse.pid.*.identifier-generation-style* above), or "krby26qt" for `storedProcGenerated` (when using base36 timestamps, as described in *dataverse.pid.*.identifier-generation-style* above).

	randomString	storedProcGenerated (sequential numbers)	storedProcGenerated (base36 timestamps)
DEPENDENT	TJ-CLKP/MLGWJC	100001/1	krby26qt/1
INDEPENDENT	MLGWJO	100002	krby27pz

As seen above, in cases where `dataverse.pid.*.identifier-generation-style` is set to `storedProcGenerated` and `dataverse.pid.*.datafile-pid-format` is set to `DEPENDENT`, each file within a dataset will be assigned a number *within* that dataset starting with "1".

Otherwise, if `dataverse.pid.*.datafile-pid-format` is set to `INDEPENDENT`, each file within the dataset is assigned with a new PID which is the next available identifier provided from the database stored procedure. In our example: "100002" when using sequential numbers or "krby27pz" when using base36 timestamps.

dataverse.pid.*.managed-list

dataverse.pid.*.excluded-list

With at least some PID services, it is possible for the authority(permission) to manage specific individual PIDs to be transferred between accounts. To handle these cases, the individual PIDs, written in the standard format, e.g. doi:10.5072/FK2ABCDEF can be added to the comma-separated `managed` or `excluded` list for a given provider. For entries on the `managed- list`, Dataverse will assume this PID Provider/account can update the metadata and landing URL for the PID at the service provider (even though it does not match the provider's authority/shoulder settings). Conversely, Dataverse will assume that PIDs on the `excluded-list` cannot be managed/updated by this provider (even though they match the provider's authority/shoulder settings). These settings are optional with the default assumption that these lists are empty.

DataCite-specific Settings

dataverse.pid.*.datacite.mds-api-url

dataverse.pid.*.datacite.rest-api-url

dataverse.pid.*.datacite.username

dataverse.pid.*.datacite.password

dataverse.feature.only-update-datacite-when-needed

PID Providers of type `datacite` require four additional parameters that define how the provider connects to DataCite. DataCite has two APIs that are used in Dataverse:

The base URL of the DataCite MDS API, used to mint and manage DOIs. Current valid values for `dataverse.pid.*.datacite.mds-api-url` are "https://mds.datacite.org" (production) and "https://mds.test.datacite.org" (testing, the default).

The DataCite REST API is also used - *PIDs API* information retrieval and *Make Data Count*. Current valid values for `dataverse.pid.*.datacite.rest-api-url` are "https://api.datacite.org" (production) and "https://api.test.datacite.org" (testing, the default).

DataCite uses HTTP Basic authentication for Fabrica and their APIs. You need to provide the same credentials (`username`, `password`) to Dataverse software to mint and manage DOIs for you. As noted above, you should use one of the more secure options for setting the password.

The *only-update-datacite-when-needed feature* flag is a global option that causes Dataverse to GET the latest metadata from DataCite for a DOI and compare it with the current metadata in Dataverse and only sending a following POST request if needed. This potentially substitutes a read for an unnecessary write at DataCite, but would result in extra reads when all metadata in Dataverse is new. Setting the flag to "true" is recommended when using DataCite file DOIs.

CrossRef-specific Settings

dataverse.pid.*.crossref.url

dataverse.pid.*.crossref.rest-api-url

dataverse.pid.*.crossref.username

dataverse.pid.*.crossref.password

dataverse.pid.*.crossref.depositor

dataverse.pid.*.crossref.depositor-email

CrossRef is an experimental provider. PID Providers of type `crossref` require six additional parameters that define how the provider connects to CrossRef. CrossRef has two APIs that are used in Dataverse:

The base URL of the CrossRef, used to mint and manage DOIs. Current valid values for `dataverse.pid.*.crossref.url` are "https://doi.crossref.org" and `dataverse.pid.*.crossref.rest-api-url` are "https://api.crossref.org" (production). `dataverse.pid.*.crossref.username=crusername dataverse.pid.*.crossref.password=secret dataverse.pid.*.crossref.depositor=xyz dataverse.pid.*.crossref.depositor-email=xyz@example.com`

CrossRef uses HTTP Basic authentication XML files can be POSTed to CrossRef where they are added to the submission queue to await processing Post URL REST API allows the search and reuse our members' metadata. Rest API and their APIs. You need to provide the same credentials (`username`, `password`) to Dataverse software to mint and manage DOIs for you. As noted above, you should use one of the more secure options for setting the password. Depositor and Depositor Email are used for the generation and distribution of Depositor reports.

EZId-specific Settings

dataverse.pid.*.ezid.api-url

dataverse.pid.*.ezid.username

dataverse.pid.*.ezid.password

Note that use of EZId is limited primarily to University of California institutions. If you have an EZId account, you will need to configure the `api-url` and your account `username` and `password`. As above, you should use one of the more secure options for setting the password.

PermaLink-specific Settings

dataverse.pid.*.permalink.base-url

dataverse.pid.*.permalink.separator

PermaLinks are a simple PID option intended for intranet and catalog use cases. They can be used without an external service or be configured with the `base-url` of a resolution service. PermaLinks also allow a custom `separator` to be used.

Note:

- If you configure `base-url`, it should include a "/" after the hostname like this: `https://demo.dataverse.org/`.
- When using multiple PermaLink providers, you should avoid ambiguous authority/separator/shoulder combinations that would result in the same overall prefix.
- Configuring PermaLink providers differing only by their separator values is not supported.
- In general, PermaLink authority/shoulder values should be alphanumeric. For other cases, admins may need to consider the potential impact of special characters in S3 storage identifiers, resolver URLs, exports, etc.

Handle-specific Settings

dataverse.pid.*.handlenet.index

dataverse.pid.*.handlenet.independent-service

dataverse.pid.*.handlenet.auth-handle

dataverse.pid.*.handlenet.key

dataverse.pid.*.handlenet.path

dataverse.pid.*.handlenet.passphrase

Note: If you are **minting your own handles** and plan to set up your own handle service, please refer to Handle.Net documentation.

Configure your Handle.net `index` to be used registering new persistent identifiers. Defaults to `300`.

Indices are used to separate concerns within the Handle system. To add data to an index, authentication is mandatory. See also chapter 1.4 "Authentication" of the Handle.Net Technical Documentation

Handle.Net servers use a public key authentication method where the public key is stored in a handle itself and the matching private key is provided from this file. Typically, the absolute path ends like `handle/svr_1/admpriv.bin`.

The key file may (and should) be encrypted with a passphrase (used for encryption with AES-128). See also chapter 1.4 "Authentication" of the Handle.Net Technical Documentation

Provide an absolute `key.path` to a private key file authenticating requests to your Handle.Net server.

Provide a `key.passphrase` to decrypt the private key file at `dataverse.pid.*.handlenet.key.path`.

Set `independent-service` to true if you want to use a Handle service which is setup to work 'independently' (No communication with the Global Handle Registry). By default this setting is false.

Set `auth-handle` to <prefix>/<suffix> to be used on a global handle service when the public key is NOT stored in the default handle. This setting is optional. If the public key is, for instance, stored in handle: `21.T12996/USER01`, `auth-handle` should be set to this value.

Backward-compatibility for Single PID Provider Installations

While using the PID Provider configuration settings described above is recommended, Dataverse installations only using a single PID Provider can use the settings below instead. In general, these legacy settings mirror those above except for not including a PID Provider id.

Configuring Your Dataverse Installation for a Single DOI Provider

Here are the configuration options for DOIs.:

JVM Options for DataCite:

- *Legacy Single PID Provider: dataverse.pid.datacite.mds-api-url*
- *Legacy Single PID Provider: dataverse.pid.datacite.rest-api-url*
- *Legacy Single PID Provider: dataverse.pid.datacite.username*
- *Legacy Single PID Provider: dataverse.pid.datacite.password*

JVM Options for EZID:

As stated above, with very few exceptions (e.g. University of California), you will not be able to use this provider.

- *Legacy Single PID Provider: dataverse.pid.ezid.api-url*
- *Legacy Single PID Provider: dataverse.pid.ezid.username*
- *Legacy Single PID Provider: dataverse.pid.ezid.password*

Database Settings:

- *:DoiProvider*
- *:Protocol*
- *:Authority*
- *:Shoulder*
- *:IdentifierGenerationStyle* (optional)
- *:DataFilePIDFormat* (optional)
- *:FilePIDsEnabled* (optional, defaults to false)

Configuring Your Dataverse Installation for a Single Handle Provider

Here are the configuration options for handles. Most notably, you need to change the `:Protocol` setting, as it defaults to DOI usage.

JVM Options:

- *Legacy Single PID Provider: dataverse.pid.handlenet.key.path*
- *Legacy Single PID Provider: dataverse.pid.handlenet.key.passphrase*
- *Legacy Single PID Provider: dataverse.pid.handlenet.index*

Database Settings:

- *:Protocol*
- *:Authority*
- *:IdentifierGenerationStyle* (optional)
- *:DataFilePIDFormat* (optional)
- *:IndependentHandleService* (optional)
- *:HandleAuthHandle* (optional)

Note: If you are **minting your own handles** and plan to set up your own handle service, please refer to Handle.Net documentation.

Configuring Your Dataverse Installation for a Single PermaLink Provider

Here are the configuration options for PermaLinks:

JVM Options:

- *Legacy Single PID Provider: dataverse.pid.permalink.base-url*

Database Settings:

- *:Protocol*
- *:Authority*
- *:Shoulder*
- *:IdentifierGenerationStyle* (optional)
- *:DataFilePIDFormat* (optional)
- *:FilePIDsEnabled* (optional, defaults to false)

You must restart Payara after making changes to these settings.

6.5.5 Auth Modes: Local vs. Remote vs. Both

There are three valid configurations or modes for authenticating users to your Dataverse installation:

Local Only Auth

Out of the box, your Dataverse installation is configured in "local only" mode. The "dataverseAdmin" superuser account mentioned in the *Installation* section is an example of a local account. Internally, these accounts are called "builtin" because they are built in to the Dataverse Software application itself.

Both Local and Remote Auth

The `authenticationproviderrow` database table controls which "authentication providers" are available within a Dataverse installation. Out of the box, a single row with an id of "builtin" will be present. For each user in a Dataverse installation, the `authenticateduserlookup` table will have a value under `authenticationproviderid` that matches this id. For example, the default "dataverseAdmin" user will have the value "builtin" under `authenticationproviderid`. Why is this important? Users are tied to a specific authentication provider but conversion mechanisms are available to switch a user from one authentication provider to the other. As explained in the *Account Creation + Management* section of the User Guide, a graphical workflow is provided for end users to convert from the "builtin" authentication provider to a remote provider. Conversion from a remote authentication provider to the builtin provider can be performed by a sysadmin with access to the "admin" API. See the *Native API* section of the API Guide for how to list users and authentication providers as JSON.

Adding and enabling a second authentication provider (*Add Authentication Provider* and *Enable or Disable an Authentication Provider*) will result in the Log In page showing additional providers for your users to choose from. By default, the Log In page will show the "builtin" provider, but you can adjust this via the *:DefaultAuthProvider* configuration option. Further customization can be achieved by setting *:AllowSignUp* to "false", thus preventing users from creating local accounts via the web interface. Please note that local accounts can also be created through the API by enabling the `builtin-users` endpoint (*:BlockedApiEndpoints (Deprecated)*) and setting the `:BuiltinUsersKey` database setting (*:BuiltinUsersKey*).

To configure Shibboleth see the *Shibboleth* section and to configure OAuth see the *OAuth Login Options* section.

Remote Only Auth

As for the "Remote only" authentication mode, it means that:

- Shibboleth or OAuth has been enabled.
- `:AllowSignUp` is set to "false" to prevent users from creating local accounts via the web interface.
- `:DefaultAuthProvider` has been set to use the desired authentication provider
- The "builtin" authentication provider has been disabled (*Enable or Disable an Authentication Provider*). Note that disabling the "builtin" authentication provider means that the API endpoint for converting an account from a remote auth provider will not work. Converting directly from one remote authentication provider to another (i.e. from GitHub to Google) is not supported. Conversion from remote is always to "builtin". Then the user initiates a conversion from "builtin" to remote. Note that longer term, the plan is to permit multiple login options to the same Dataverse installation account per https://github.com/IQSS/dataverse/issues/3487 (so all this talk of conversion will be moot) but for now users can only use a single login option, as explained in the *Account Creation + Management* section of the User Guide. In short, "remote only" might work for you if you only plan to use a single remote authentication provider such that no conversion between remote authentication providers will be necessary.

6.5.6 Bearer Token Authentication

Bearer tokens are defined in RFC 6750 and can be used as an alternative to API tokens. This is an experimental feature hidden behind a feature flag.

To enable bearer tokens, you must install and configure Keycloak (for now, see *OpenID Connect (OIDC)* in the Developer Guide) and enable the *dataverse.feature.api-bearer-auth* feature flag.

You can test that bearer tokens are working by following the example under *Using Preview URL Tokens as API Token* in the API Guide.

6.5.7 SMTP/Email Configuration

The installer prompts you for some basic options to configure Dataverse to send email using your SMTP server, but in many cases, extra configuration may be necessary.

Make sure the *dataverse.mail.system-email* has been set. Email will not be sent without it. A hint will be logged about this fact. If you want to separate system email from your support team's email, take a look at *dataverse.mail.support-email*.

Then check the list of commonly used settings at the top of *dataverse.mail.mta.**.

If you have trouble, consider turning on debugging with *dataverse.mail.debug*.

6.5.8 Database Persistence

The Dataverse software uses a PostgreSQL database to store objects users create. You can configure basic and advanced settings for the PostgreSQL database connection with the help of MicroProfile Config API.

Basic Database Settings

1. Any of these settings can be set via system properties (see *JVM Options* starting at *dataverse.db.name*), environment variables or other MicroProfile Config mechanisms supported by the app server. See Payara docs for supported sources.
2. Remember to protect your secrets. See *Secure Password Storage* for more information.
3. Environment variables follow the key, replacing any dot, colon, dash, etc. into an underscore "_" and all uppercase letters. Example: `dataverse.db.host` -> `DATAVERSE_DB_HOST`

MPCONFIG Key	Description	Default
dataverse.db.host	The PostgreSQL server to connect to.	`localhost`
dataverse.db.port	The PostgreSQL server port to connect to.	`5432`
dataverse.db.user	The PostgreSQL user name to connect with.	`dataverse` (installer sets to `dvnapp`)
dataverse.db.passwo	The PostgreSQL users password to connect with. **Please note the safety advisory above.**	*No default*
dataverse.db.name	The PostgreSQL database name to use for the Dataverse installation.	`dataverse` (installer sets to `dvndb`)
dataverse.db.parame	Connection parameters, such as `sslmode=require`. See Postgres JDBC docs Note: you don't need to provide the initial "?".	*Empty string*

Advanced Database Settings

The following options are useful in many scenarios. You might be interested in debug output during development or monitoring performance in production.

You can find more details within the Payara docs:

- User Guide: Connection Pool Configuration
- Tech Doc: Advanced Connection Pool Configuration.

Connection Validation

MPCONFIG Key	Description	Default
dataverse.db.is-connection-validation-required	`true`: Validate connections, allow server to reconnect in case of failure.	false
dataverse.db.cor validation-method	The method of connection validation: `table|autocommit|meta-data|custom-validation`.	*Empty string*
dataverse.db.val table-name	The name of the table used for validation if the validation method is set to `table`.	*Empty string*
dataverse.db.val classname	The name of the custom class used for validation if the `validation-method` is set to `custom-validation`.	*Empty string*
dataverse.db.val atmost-once-period-in-seconds	Specifies the time interval in seconds between successive requests to validate a connection at most once.	`0` (disabled)

Connection & Statement Leaks

MPCONFIG Key	Description	Default
dataverse.db.cor leak-timeout-in-seconds	Specify timeout when connections count as "leaked".	`0` (disabled)
dataverse.db.cor leak-reclaim	If enabled, leaked connection will be reclaimed by the pool after connection leak timeout occurs.	`false`
dataverse.db.sta leak-timeout-in-seconds	Specifiy timeout when statements should be considered to be "leaked".	`0` (disabled)
dataverse.db.sta leak-reclaim	If enabled, leaked statement will be reclaimed by the pool after statement leak timeout occurs.	`false`

Logging & Slow Performance

MPCONFIG Key	Description	Default
dataverse.db.stat timeout-in-seconds	Timeout property of a connection to enable termination of abnormally long running queries.	-1 (disabled)
dataverse.db.slo query-threshold-in-seconds	SQL queries that exceed this time in seconds will be logged.	-1 (disabled)
dataverse.db.log jdbc-calls	When set to true, all JDBC calls will be logged allowing tracing of all JDBC interactions including SQL.	`false`

Database Configuration Tips

In this section you can find some example scenarios of advanced configuration for the database connection that can improve service performance and availability.

Database Connection Recovery

Consider the following scenario: if there is no advanced configuration for the database connection and the Dataverse server loses that connection, for example if the database host is down, the server will be "dead" even after the database server is back to normal. The only solution to recover Dataverse would be to restart the service. To avoid this situation, the following settings can be used to configure validation of the database connection. This way, the database connection can be automatically recovered after a failure, improving the server availability. For a Docker installation, it is suggested to create an init.d script so that if the container needs to be recreated, these settings will always be configured.

```
# Enable database connection validation
asadmin create-jvm-options "-Ddataverse.db.is-connection-validation-required=true"
# Configure to use a database table as the validation method
asadmin create-jvm-options "-Ddataverse.db.connection-validation-method=table"
# Configure the "setting" table to be used for connection validation, but any tables can␣
↪be used
asadmin create-jvm-options "-Ddataverse.db.validation-table-name=setting"
# Configure a validation period of 60 seconds, but different values may be used
asadmin create-jvm-options "-Ddataverse.db.validate-atmost-once-period-in-seconds=60"
```

6.5.9 File Storage

By default, a Dataverse installation stores all data files (files uploaded by end users) on the filesystem at `/usr/local/payara7/glassfish/domains/domain1/files`. This path can vary based on answers you gave to the installer (see the *Running the Dataverse Software Installer* section of the Installation Guide) or afterward by reconfiguring the `dataverse.files.\<id\>.directory` JVM option described below.

A Dataverse installation can alternately store files in a Swift or S3-compatible object store, or on a Globus endpoint, and can now be configured to support multiple stores at once. A Dataverse installation may also be configured to reference some files (e.g. large and/or sensitive data) stored in a web or Globus accessible trusted remote store. With a

multi-store configuration, the location for new files can be controlled on a per-Dataverse collection or per-dataset basis. *Scaling Dataverse with Data Size* provides more detail about the pros and cons of different types of storage.

A Dataverse installation can be configured to allow out of band upload by setting the `dataverse.files.\<id\>.upload-out-of-band` JVM option to `true`. By default, Dataverse supports uploading files via the *Add a File to a Dataset*. With S3 stores, a direct upload process can be enabled to allow sending the file directly to the S3 store (without any intermediate copies on the Dataverse server). With the upload-out-of-band option enabled, it is also possible for file upload to be managed manually or via third-party tools, with the *Adding the Uploaded file to the Dataset* API call (described in the *Direct DataFile Upload/Replace API* page) used to add metadata and inform Dataverse that a new file has been added to the relevant store.

The following sections describe how to set up various types of stores and how to configure for multiple stores. See also *Storage: Choosing the Right Store*.

Multi-store Basics

To support multiple stores, a Dataverse installation now requires an id, type, and label for each store (even for a single store configuration). These are configured by defining two required jvm options:

```
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.<id>.type=<type>"
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.<id>.label=<label>"
```

Out of the box, a Dataverse installation is configured to use local file storage in the 'file' store by default. You can add additional stores and, as a superuser, configure specific Dataverse collections to use them (by editing the 'General Information' for the Dataverse collection as described in the *Managing Datasets and Dataverse Collections* section).

Note that the "-Ddataverse.files.directory", if defined, continues to control where temporary files are stored (in the /temp subdir of that directory), independent of the location of any 'file' store defined above. (See also the option reference: *dataverse.files.directory*)

If you wish to change which store is used by default, you'll need to delete the existing default storage driver and set a new one using jvm options.

```
./asadmin $ASADMIN_OPTS delete-jvm-options "-Ddataverse.files.storage-driver-id=file"
./asadmin $ASADMIN_OPTS create-jvm-options "-Ddataverse.files.storage-driver-id=<id>"
```

It is also possible to set maximum file upload size limits per store. See the *:MaxFileUploadSizeInBytes* setting below.

Labels for File Stores

If you find yourself adding many file stores with various configurations such as per-file limits and direct upload, you might find it helpful to make the label descriptive.

For example, instead of simply labeling an S3 store as "S3"…

```
./asadmin create-jvm-options "\-Ddataverse.files.s3xl.label=S3"
```

… you might want to include some extra information such as the example below.

```
./asadmin create-jvm-options "\-Ddataverse.files.s3xl.label=S3XL, Filesize limit: 100GB,␣
↪direct-upload"
```

Please keep in mind that the UI will only show so many characters, so labels are best kept short.

File Storage

File stores have one option - the directory where files should be stored. This can be set using

```
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.<id>.directory=<file␣
↪directory>"
```

Multiple file stores should specify different directories (which would nominally be the reason to use multiple file stores), but one may share the same directory as "-Ddataverse.files.directory" option - this would result in temp files being stored in the /temp subdirectory within the file store's root directory.

See also *File Stores*.

Swift Storage

Rather than storing data files on the filesystem, you can opt for an experimental setup with a Swift Object Storage backend. Each dataset that users create gets a corresponding "container" on the Swift side, and each data file is saved as a file within that container.

In order to configure a Swift installation, you need to complete these steps to properly modify the JVM options:

First, run all the following create commands with your Swift endpoint information and credentials:

```
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.<id>.type=swift"
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.<id>.
↪defaultEndpoint=endpoint1"
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.<id>.authType.
↪endpoint1=your-auth-type"
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.<id>.authUrl.
↪endpoint1=your-auth-url"
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.<id>.tenant.
↪endpoint1=your-tenant-name"
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.<id>.username.
↪endpoint1=your-username"
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.<id>.endpoint.
↪endpoint1=your-swift-endpoint"
```

`auth_type` can either be `keystone`, `keystone_v3`, or it will assumed to be `basic`. `auth_url` should be your keystone authentication URL which includes the tokens (e.g. for keystone, `https://openstack.example.edu:35357/v2.0/tokens` and for keystone_v3, `https://openstack.example.edu:35357/v3/auth/tokens`). `swift_endpoint` is a URL that looks something like `https://rdgw.swift.example.org/swift/v1`.

Then create a password alias by running (without changes):

```
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.swift.password.endpoint1=
↪'\${ALIAS=swiftpassword-alias}'"
./asadmin $ASADMIN_OPTS create-password-alias swiftpassword-alias
```

The second command will trigger an interactive prompt asking you to input your Swift password.

Note: you may choose a different way to secure this password, depending on your use case. See *Secure Password Storage* for more options.

Second, update the JVM option `dataverse.files.storage-driver-id` by running the delete command:

```
./asadmin $ASADMIN_OPTS delete-jvm-options "\-Ddataverse.files.storage-driver-id=file"
```

Then run the create command:

```
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.storage-driver-id=swift"
```

You also have the option to set a **custom container name separator.** It is initialized to _, but you can change it by running the create command:

```
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.swift.
folderPathSeparator=-"
```

By default, your Swift installation will be public-only, meaning users will be unable to put access restrictions on their data. If you are comfortable with this level of privacy, the final step in your setup is to set the *:PublicInstall* setting to *true*.

In order to **enable file access restrictions**, you must enable Swift to use temporary URLs for file access. To enable usage of temporary URLs, set a hash key both on your swift endpoint and in your swift.properties file. You can do so by running the create command:

```
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.swift.hashKey.
endpoint1=your-hash-key"
```

You also have the option to set a custom expiration length, in seconds, for a generated temporary URL. It is initialized to 60 seconds, but you can change it by running the create command:

```
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.swift.
temporaryUrlExpiryTime=3600"
```

In this example, you would be setting the expiration length for one hour.

Setting up Compute with Swift

Once you have configured a Swift Object Storage backend, you also have the option of enabling a connection to a computing environment. To do so, you need to configure the database settings for *:ComputeBaseUrl* and *:CloudEnvironmentName*.

Once you have set up `:ComputeBaseUrl` properly in both the Dataverse installation and your cloud environment, validated users will have three options for accessing the computing environment:

- Compute on a single dataset
- Compute on multiple datasets
- Compute on a single datafile

The compute tool options on dataset and file pages will link validated users to your computing environment. If a user is computing on one dataset, the compute tool option will redirect to:

```
:ComputeBaseUrl?datasetPersistentId
```

If a user is computing on multiple datasets, the compute tool option will redirect to:

```
:ComputeBaseUrl/multiparty?datasetPersistentId&anotherDatasetPersistentId&anotherDatasetPersistentId&.
..
```

If a user is computing on a single file, depending on the configuration of your installation, the compute tool option will either redirect to:

```
:ComputeBaseUrl?datasetPersistentId=yourObject
```

if your installation's *:PublicInstall* setting is true, or:

```
:ComputeBaseUrl?datasetPersistentId=yourObject&temp_url_sig=yourTempUrlSig&temp_url_expires=yourTempUrl
```

You can configure this redirect properly in your cloud environment to generate a temporary URL for access to the Swift objects for computing.

Amazon S3 Storage (or Compatible)

The Dataverse Software supports Amazon S3 storage as well as other S3-compatible stores (like Ceph RADOS S3 Gateway and many more) for files uploaded to your Dataverse installation.

The Dataverse Software S3 driver supports multi-part upload for large files (over 1 GB by default - see the min-part-size option in the table below to change this).

Note: The Dataverse Project Team is most familiar with AWS S3, and can provide support on its usage with the Dataverse Software. Thanks to community contributions, the application's architecture also allows non-AWS S3 providers. The Dataverse Project Team can provide very limited support on these other providers. We recommend reaching out to the wider Dataverse Project Community if you have questions.

See also *S3 Stores: Object Storage via S3*.

First: Set Up Accounts and Access Credentials

Note: As of version 5.14, if Dataverse is running in an EC2 instance it will prefer Role-Based Access Control over the S3 default profile, even if administrators configure Dataverse with programmatic access keys. Named profiles can still be used to override RBAC for specific datastores. RBAC is preferential from a security perspective as there are no keys to rotate or have stolen. If you intend to assign a role to your EC2 instance, you will still need the `~/.aws/config` file to specify the region but you need not generate credentials for the default profile. For more information please see https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html

The Dataverse Software and the AWS SDK make use of the "AWS credentials profile file" and "AWS config profile file" located in `~/.aws/` where ~ is the home directory of the user you run Payara as. This file can be generated via either of two methods described below:

1. Manually through creation of the credentials and config files or
2. Automatically via the AWS console commands.

Some usage scenarios might be eased without generating these files. You may also provide *static credentials via MicroProfile Config*, see below.

Preparation When Using Amazon's S3 Service

You'll need an AWS account with an associated S3 bucket for your installation to use. From the S3 management console (e.g. https://console.aws.amazon.com/), you can poke around and get familiar with your bucket.

Make note of the **bucket's name** and the **region** its data is hosted in.

To **create a user** with full S3 access and nothing more for security reasons, we recommend using IAM (Identity and Access Management). See IAM User Guide for more info on this process.

To use programmatic access, **Generate the user keys** needed for a Dataverse installation afterwards by clicking on the created user.

Preparation When Using Custom S3-Compatible Service

We assume you have your S3-compatible custom storage in place, up and running, ready for service.

Please make note of the following details:

- **Endpoint URL** - consult the documentation of your service on how to find it.
  - Example: http://localhost.localstack.cloud:4566
- **Region:** Optional, but some services might use it. Consult your service documentation.
  - Example: *us-east-1*
- **Access key ID and secret access key:** Usually you can generate access keys within the user profile of your service.
  - Example:
    - ID: *Q3AM3UQ867SPQQA43P2F*
    - Key: *zuf+tfteSlswRu7BJ86wekitnifILbZam1KYY3TG*
- **Bucket name:** The Dataverse installation will fail opening and uploading files on S3 if you don't create one.
  - Example: *dataverse*

Manually Set Up Credentials File

To create the `~/.aws/credentials` file manually, you will need to generate a key/secret key (see above). Once you have acquired the keys, they need to be added to the `credentials` file. The format for credentials is as follows:

```
[default]
aws_access_key_id = <insert key, no brackets>
aws_secret_access_key = <insert secret key, no brackets>
```

While using Amazon's service, you must also specify the AWS region in the `~/.aws/config` file, for example:

```
[default]
region = us-east-1
```

Additional profiles can be added to these files by appending the relevant information in additional blocks:

```
[default]
aws_access_key_id = <insert key, no brackets>
aws_secret_access_key = <insert secret key, no brackets>

[profilename2]
aws_access_key_id = <insert key, no brackets>
aws_secret_access_key = <insert secret key, no brackets>
```

Place these two files in a folder named `.aws` under the home directory for the user running your Dataverse Installation on Payara. (From the AWS Command Line Interface Documentation: "In order to separate credentials from less sensitive options, region and output format are stored in a separate file named config in the same folder")

Console Commands to Set Up Access Configuration

Begin by installing the CLI tool pip (package installer for Python) to install the AWS command line interface if you don't have it.

First, we'll get our access keys set up. If you already have your access keys configured, skip this step. From the command line, run:

- `pip install awscli`
- `aws configure`

You'll be prompted to enter your Access Key ID and secret key, which should be issued to your AWS account. The subsequent config steps after the access keys are up to you. For reference, the keys will be stored in `~/.aws/credentials`, and your AWS access region in `~/.aws/config`.

TIP: When using a custom S3 URL endpoint, you need to add it to every `aws` call: `aws --endpoint-url <URL> s3 ...`
(you may omit it while configuring).

Second: Configure Your Dataverse Installation to use S3 Storage

To set up an S3 store, you must define the id, type, and label as for any store:

```
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.<id>.type=s3"
./asadmin $ASADMIN_OPTS create-jvm-options "\-Ddataverse.files.<id>.label=<label>"
```

Then, we'll need to identify which S3 bucket we're using. Replace <your_bucket_name> with, of course, your bucket:

```
./asadmin create-jvm-options "-Ddataverse.files.<id>.bucket-name=<your_bucket_name>"
```

Optionally, you can have users download files from S3 directly rather than having files pass from S3 through Payara to your users. To accomplish this, set `dataverse.files.<id>.download-redirect` to `true` like this:

```
./asadmin create-jvm-options "-Ddataverse.files.<id>.download-redirect=true"
```

If you enable `dataverse.files.<id>.download-redirect` as described above, note that the S3 URLs expire after an hour by default but you can configure the expiration time using the `dataverse.files.<id>.url-expiration-minutes` JVM option. Here's an example of setting the expiration time to 120 minutes:

```
./asadmin create-jvm-options "-Ddataverse.files.<id>.url-expiration-minutes=120"
```

By default, your store will use the [default] profile in you .aws configuration files. To use a different profile, which would be necessary if you have two s3 stores at different locations, you can specify the profile to use:

```
./asadmin create-jvm-options "-Ddataverse.files.<id>.profile=<profilename>"
```

Larger installations may want to increase the number of open S3 connections allowed (default is 256): For example,

```
./asadmin create-jvm-options "-Ddataverse.files.<id>.connection-pool-size=4096"
```

S3 Tagging

By default, when direct upload to an S3 store is configured, Dataverse will place a `temp` tag on the file being uploaded for an easier cleanup in case the file is not added to the dataset after upload (e.g., if the user cancels the operation). (See *S3 Tags and Direct Upload*.) If your S3 store does not support tagging and gives an error when direct upload is configured, you can disable the tagging by using the `dataverse.files.<id>.disable-tagging` JVM option. For example:

```
./asadmin create-jvm-options "-Ddataverse.files.<id>.disable-tagging=true"
```

Disabling the `temp` tag makes it harder to identify abandoned files that are not used by your Dataverse instance (i.e. one cannot search for the `temp` tag in a delete script). These should still be removed to avoid wasting storage space. To clean up these files and any other leftover files, regardless of whether the `temp` tag is applied, you can use the *Cleanup Storage of a Dataset* API endpoint.

Note that if you disable tagging, you should should omit the `x-amz-tagging:dv-state=temp` header when using the *Direct DataFile Upload/Replace API*, as noted in that section.

Finalizing S3 Configuration

In case you would like to configure Dataverse to use a custom S3 service instead of Amazon S3 services, please add the options for the custom URL and region as documented below. Please read above if your desired combination has been tested already and what other options have been set for a successful integration.

Lastly, go ahead and restart your Payara server. With Dataverse deployed and the site online, you should be able to upload datasets and data files and see the corresponding files in your S3 bucket. Within a bucket, the folder structure emulates that found in local file storage.

List of S3 Storage Options

JVM Option	Value	Description	Default value
dataverse.files.storage-driver-id	<id>	Enable <id> as the default storage driver.	`file`
dataverse.files.<id>.type	`s3`	**Required** to mark this storage as S3 based.	(none)
dataverse.files.<id>.label	<?>	**Required** label to be shown in the UI for this storage	(none)
dataverse.files.<id>.bucket-name	<?>	The bucket name. See above.	(none)
dataverse.files.<id>.download-redirect	`true/fal`	Enable direct download or proxy through Dataverse.	`false`
dataverse.files.<id>.upload-redirect	`true/fal`	Enable direct upload of files added to a dataset in the S3 store.	`false`
dataverse.files.<id>.upload-out-of-band	`true/fal`	Allow upload of files by out-of-band methods (using some tool other than Dataverse)	`false`
dataverse.files.<id>.ingestsizelimit	<size in bytes>	Maximum size of directupload files that should be ingested	(none)
dataverse.files.<id>.url-expiration-minutes	<?>	If direct uploads/downloads: time until links expire. Optional.	60
dataverse.files.<id>.min-part-size	<?>	Multipart direct uploads will occur for files larger than this. Optional.	`1024**3`
dataverse.files.<id>.custom-endpoint-url	<?>	Use custom S3 endpoint. Needs URL either with or without protocol.	(none)
dataverse.files.<id>.custom-endpoint-region	<?>	Only used when using custom endpoint. Optional.	`dataverse`
dataverse.files.<id>.profile	<?>	Allows the use of AWS profiles for storage spanning multiple AWS accounts.	(none)
dataverse.files.<id>.proxy-url	<?>	URL of a proxy protecting the S3 store. Optional.	(none)
dataverse.files.<id>.path-style-access	`true/fal`	Use path style buckets instead of subdomains. Optional.	`false`
dataverse.files.<id>.chunked-encoding	`true/fal`	Disable chunked encoding. Optional	`true`
dataverse.files.<id>.connection-pool-size	<?>	The maximum number of open connections to the S3 server	`256`
dataverse.files.<id>.disable-tagging	`true/fal`	Do not place the `temp` tag when redirecting the upload to the S3 server.	`false`
dataverse.files.<id>.disable-multipart-download-for-indirect-download	`true/fal`	Disable multipart download for indirect downloads from S3.	`false`

MicroProfile Config Option	Value	Description	Default value
dataverse.files.<id>.access-key	<?>	*Provide static access key ID. Read before use!*	""
dataverse.files.<id>.secret-key	<?>	*Provide static secret access key. Read before use!*	""

Credentials via MicroProfile Config

Optionally, you may provide static credentials for each S3 storage using MicroProfile Config options:

- `dataverse.files.<id>.access-key` for this storage's "access key ID"
- `dataverse.files.<id>.secret-key` for this storage's "secret access key"

You may provide the values for these via any supported MicroProfile Config API source.

WARNING: *For security, do not use the sources "environment variable" or "system property" (JVM option) in a production context! Rely on password alias, secrets directory or cloud based sources as described at Secure Password Storage instead!*

NOTE:

1. Providing both AWS CLI profile files (as setup in first step) and static keys, credentials from `~/.aws` will win over configured keys when valid!
2. A non-empty `dataverse.files.<id>.profile` will be ignored when no credentials can be found for this profile name. Current codebase does not make use of "named profiles" as seen for AWS CLI besides credentials.

Reported Working S3-Compatible Storage

Ceph Object Gateway (added July 2026/Dataverse v6.12) Set `dataverse.files.<id>.disable-multipart-download-for-indirect-download=true` if not using direct download. (This forces the S3 server to handle part reassembly and avoid incompatible headers that cause *412* errors from the Ceph Gateway.)

StorJ Object Store
: StorJ is a distributed object store that can be configured with an S3 gateway. Per the S3 Storage instructions above, you'll first set up the StorJ S3 store by defining the id, type, and label. After following the general installation, set the following configuration to use a StorJ object store: `dataverse.files.<id>.chunked-encoding=false`. For step-by-step instructions see https://docs.storj.io/dcs/how-tos/dataverse-integration-guide/

  Note that for direct uploads and downloads, Dataverse redirects to the proxy-url but presigns the urls based on the `dataverse.files.<id>.custom-endpoint-url`. Also, note that if you choose to enable `dataverse.files.<id>.download-redirect` the S3 URLs expire after 60 minutes by default. You can change that minute value to reflect a timeout value that's more appropriate by using `dataverse.files.<id>.url-expiration-minutes`.

Surf Object Store v2019-10-30
: Set `dataverse.files.<id>.chunked-encoding=false` and `dataverse.files.<id>.path-style-request=true` to use Surf Object Store. You will need the Swift client (documented at <http://doc.swift.surfsara.nl/en/latest/Pages/Clients/s3cred.html>) to create the access key and secret key for the S3 interface.

Note that the `dataverse.files.<id>.proxy-url` setting can be used in installations where the object store is proxied, but it should be considered an advanced option that will require significant expertise to properly configure. For direct uploads and downloads, Dataverse redirects to the proxy-url but presigns the urls based on the `dataverse.files.<id>.custom-endpoint-url`. Additional configuration (appropriate CORS settings, proxy caching/timeout configuration, and proxy settings to pass headers to/from S3 and to avoid adding additional headers) will also be needed to enable use of a proxy with direct upload and download. For Amazon AWS, see comments in the edu.harvard.iq.dataverse.dataaccess.S3AccessIO class about support for AWS's bucket-specific DNS names.

SeaweedFS
: SeaweedFS is a distributed storage system that has S3 compatibility. Set the S3 storage options as explained above. Make sure to set `dataverse.files.<id>.path-style-access` to `true`. You will need to create the bucket beforehand. You can do this with the filer API using curl commands. For example, to create an empty bucket called `dataverse`:

```
curl -X POST "http://localhost:8888/buckets/"
curl -X POST "http://localhost:8888/buckets/dataverse/"
```

You will also need to set an access and secret key. One way to do this is via a static file. As an example, your `config.json` might look like this if you're using a bucket called `dataverse`:

```
{
  "identities": [
    {
      "name": "anonymous",
      "credentials": [
        {
          "accessKey": "secret",
          "secretKey": "secret"
        }
      ],
      "actions": [
        "Read:dataverse",
        "List:dataverse",
        "Tagging:dataverse",
        "Write:dataverse"
      ]
    }
  ]
}
```

And lastly, to start up the SeaweedFS server and various components you could use a command like this:

```
weed server -s3 -metricsPort=9327 -dir=/data -s3.config=/config.json
```

VAST DataStore

VAST DataStore must be configured with an S3 gateway. A Dataverse bucket must be created. Follow VAST DataStore documentation to configure the S3 gateway. Set `dataverse.files.<id>.path-style-access=true` since VAST DataStore uses path style access.

Additional Reported Working S3-Compatible Storage

If you are successfully using an S3 storage implementation not yet listed above, please feel free to open an issue at Github and describe your setup. We will be glad to add it.

Migrating from Local Storage to S3

Is currently documented on the *Deployment* page.

Trusted Remote Storage

In addition to having the type "remote" and requiring a label, Trusted Remote Stores are defined in terms of a baseURL - all files managed by this store must be at a path starting with this URL, and a baseStore - a file, s3, or swift store that can be used to store additional ancillary dataset files (e.g. metadata exports, thumbnails, auxiliary files, etc.). These and other available options are described in the table below.

Trusted remote stores can range from being a static trusted website to a sophisticated service managing access requests and logging activity and/or managing access to a secure enclave. See *Scaling Dataverse with Data Size* (specifically *Remote Stores*) and *Big Data Support* for additional information on how to use a trusted remote store. For specific remote stores, consult their documentation when configuring the remote store in your Dataverse installation.

Note that in the current implementation, activities where Dataverse needs access to data bytes, e.g. to create thumbnails or validate hash values at publication will fail if a remote store does not allow Dataverse access. Implementers of such trusted remote stores should consider using Dataverse's settings to disable ingest, validation of files at publication, etc. as needed.

Once you have configured a trusted remote store, you can point your users to the *Add a Remote File to a Dataset* section of the API Guide.

JVM Option	Value	Description	Default value
dataverse.files.<id>.type	`remote`	**Required** to mark this storage as remote.	(none)
dataverse.files.<id>.label	<?>	**Required** label to be shown in the UI for this storage.	(none)
dataverse.files.<id>.base-url	<?>	**Required** All files must have URLs of the form <baseUrl>/* .	(none)
dataverse.files.<id>.base-store	<?>	**Required** The id of a base store (of type file, s3, or swift).	(the default store)
dataverse.files.<id>.upload-out-of-band	`true`	**Required to be true** Dataverse does not manage file placement	`false`
dataverse.files.<id>.download-redirect	`true/fals`	Enable direct download (should usually be true).	`false`
dataverse.files.<id>.secret-key	<?>	A key used to sign download requests sent to the remote store. Optional.	(none)
dataverse.files.<id>.public	`true/fals`	True if the remote store does not enforce Dataverse access controls	`false`
dataverse.files.<id>.ingestsizelimit	<size in bytes>	Maximum size of files that should be ingested	(none)
dataverse.files.<id>.url-expiration-minutes	<?>	If direct downloads and using signing: time until links expire. Optional.	60
dataverse.files.<id>.remote-store-name	<?>	A short name used in the UI to indicate where a file is located. Optional.	(none)
dataverse.files.<id>.remote-store-url	<?>	A URL to an info page about the remote store used in the UI. Optional.	(none)
dataverse.files.<id>.files-not-accessible-by-dataverse	`true/fals`	True if the file is at the URL provided, false if that is a landing page	`false`

Globus Storage

Globus stores allow Dataverse to manage files stored in Globus endpoints or to reference files in remote Globus endpoints, with users leveraging Globus to transfer files to/from Dataverse (rather than using HTTP/HTTPS). See *Big Data Support* for additional information on how to use a globus store. Consult the Globus documentation for information about using Globus and configuring Globus endpoints.

In addition to having the type "globus" and requiring a label, Globus Stores share many options with Trusted Remote Stores and options to specify and access a Globus endpoint(s). As with Remote Stores, Globus Stores also use a baseStore - a file, s3, or swift store that can be used to store additional ancillary dataset files (e.g. metadata exports, thumbnails, auxiliary files, etc.). These and other available options are described in the table below.

There are two types of Globus stores:

- managed - where Dataverse manages the Globus endpoint, deciding where transferred files are stored and managing access control for users transferring files to/from Dataverse
- remote - where Dataverse references files that remain on trusted remote Globus endpoints

A managed Globus store connects to standard/file-based Globus endpoint. It is also possible to configure an S3 store as a managed store, if the managed endpoint uses an underlying S3 store via the Globus S3 Connector. With the former, Dataverse has no direct access to the file contents and functionality related to ingest, fixity hash validation, etc. are not available. With the latter, Dataverse can access files internally via S3 and the functionality supported is similar to that when using S3 direct upload.

Once you have configured a globus store, or configured an S3 store for Globus access, it is recommended that you install the dataverse-globus app to allow transfers in/out of Dataverse to be initated via the Dataverse user interface. Alternately, you can point your users to the *Globus Transfer API* for information about API support.

JVM Option	Value	Description	Default value
dataverse.files.<id>.type	`globus`	**Required** to mark this storage as globus enabled.	(none)
dataverse.files.<id>.label	<?>	**Required** label to be shown in the UI for this storage.	(none)
dataverse.files.<id>.base-store	<?>	**Required** The id of a base store (of type file, s3, or swift).	(the default store)
dataverse.files.<id>.remote-store-name	<?>	A short name used in the UI to indicate where a file is located. Optional.	(none)
dataverse.files.<id>.remote-store-url	<?>	A url to an info page about the remote store used in the UI. Optional.	(none)
dataverse.files.<id>.managed	`true/f`	Whether dataverse manages an associated Globus endpoint	`false`
dataverse.files.<id>.transfer-endpoint-with-basepath	<?>	The *managed* Globus endpoint id and associated base path for file storage	(none)
dataverse.files.<id>.globus-token	<?>	A Globus token (base64 endcoded <Globus user id>:<Credential> for a managed store) - using a minipro-file alias is recommended	(none)
dataverse.files.<id>.referenc endpoints-with-basepaths	<?>	A comma separated list of *remote* trusted Globus endpoint id/<basePath>s	(none)
dataverse.files.<id>.files-not-accessible-by-dataverse	`true/f`	Should be false for S3 Connector-based *managed* stores, true for others	`false`
dataverse.files.<id>.public	`true/f`	True can be used to disable users ability restrict/embargo files	`false`

Temporary Upload File Storage

When uploading files via the API or Web UI, you need to be aware that multiple steps are involved to enable features like ingest processing, transfer to a permanent storage, checking for duplicates, unzipping etc.

All of these processes are triggered after finishing transfers over the wire and moving the data into a temporary (configurable) location on disk at *${dataverse.files.directory}*`/temp`.

Before being moved there,

- JSF Web UI uploads are stored at *${dataverse.files.uploads}*, defaulting to `/usr/local/payara7/glassfish/domains/domain1/uploads` folder in a standard installation. This place is configurable and might be set to a separate disk volume where stale uploads are purged periodically.
- API uploads are stored at the system's temporary files location indicated by the Java system property `java.io.tmpdir`, defaulting to `/tmp` on Linux. If this location is backed by a tmpfs on your machine, large file uploads via API will cause RAM and/or swap usage bursts. You might want to point this to a different location, restrict maximum size of it, and monitor for stale uploads.

6.5.10 Rate Limiting

Rate limiting has been added to prevent users from over taxing the system either deliberately or by runaway automated processes. Rate limiting can be configured on a tier level with tier 0 being reserved for guest users and tiers 1-any for authenticated users. Superuser accounts are exempt from rate limiting. Rate limits can be imposed on command APIs by configuring the tier, the command, and the hourly limit in the database. Two database settings configure the rate limiting. Note: If either of these settings exist in the database rate limiting will be enabled (note that a Payara restart is required for the setting to take effect). If neither setting exists rate limiting is disabled.

- :RateLimitingDefaultCapacityTiers is the number of calls allowed per hour if the specific command is not configured. The values represent the number of calls per hour per user for tiers 0,1,... A value of -1 can be used to signify no rate limit. Tiers not specified in this setting will default to *-1* (No Limit). I.e., -d "10000" is equivalent to -d "10000,-1,-1,..."

```
curl http://localhost:8080/api/admin/settings/:RateLimitingDefaultCapacityTiers -X PUT -
↪d '10000,20000'
```

- :RateLimitingCapacityByTierAndAction is a JSON object specifying the rate by tier and a list of actions (commands). This allows for more control over the rate limit of individual API command calls. In the following example, calls made by a guest user (tier 0) for API GetLatestPublishedDatasetVersionCommand is further limited to only 10 calls per hour, while an authenticated user (tier 1) will be able to make 30 calls per hour to the same API.

`rate-limit-actions.json` Example JSON for RateLimitingCapacityByTierAndAction

```
curl http://localhost:8080/api/admin/settings/:RateLimitingCapacityByTierAndAction -X␣
↪PUT -d '[{"tier": 0, "limitPerHour": 10, "actions": [
↪"GetLatestPublishedDatasetVersionCommand", "GetPrivateUrlCommand", "GetDatasetCommand",
↪ "GetLatestAccessibleDatasetVersionCommand"]}, {"tier": 0, "limitPerHour": 1, "actions
↪": ["CreateGuestbookResponseCommand", "UpdateDatasetVersionCommand",
↪"DestroyDatasetCommand", "DeleteDataFileCommand", "FinalizeDatasetPublicationCommand",
↪"PublishDatasetCommand"]}, {"tier": 1, "limitPerHour": 30, "actions": [
↪"CreateGuestbookResponseCommand", "GetLatestPublishedDatasetVersionCommand",
↪"GetPrivateUrlCommand", "GetDatasetCommand", "GetLatestAccessibleDatasetVersionCommand
↪", "UpdateDatasetVersionCommand", "DestroyDatasetCommand", "DeleteDataFileCommand",
↪"FinalizeDatasetPublicationCommand", "PublishDatasetCommand"]}]'
```

6.5.11 Branding Your Installation

A Dataverse installation can be branded in a number of ways.

A simple option for branding your installation is to adjust the theme of a Dataverse collection. You can change colors, add a logo, add a tagline, or add a website link to the Dataverse collection header section of the page. These options are outlined under *Theme* in the *Dataverse Collection Management* section of the User Guide.

More advanced customization is described below and covers the following areas.

- Custom installation name/brand
- Custom header
- Navbar settings
- Custom welcome/homepage
- Custom footer
- Footer settings
- CSS stylesheet

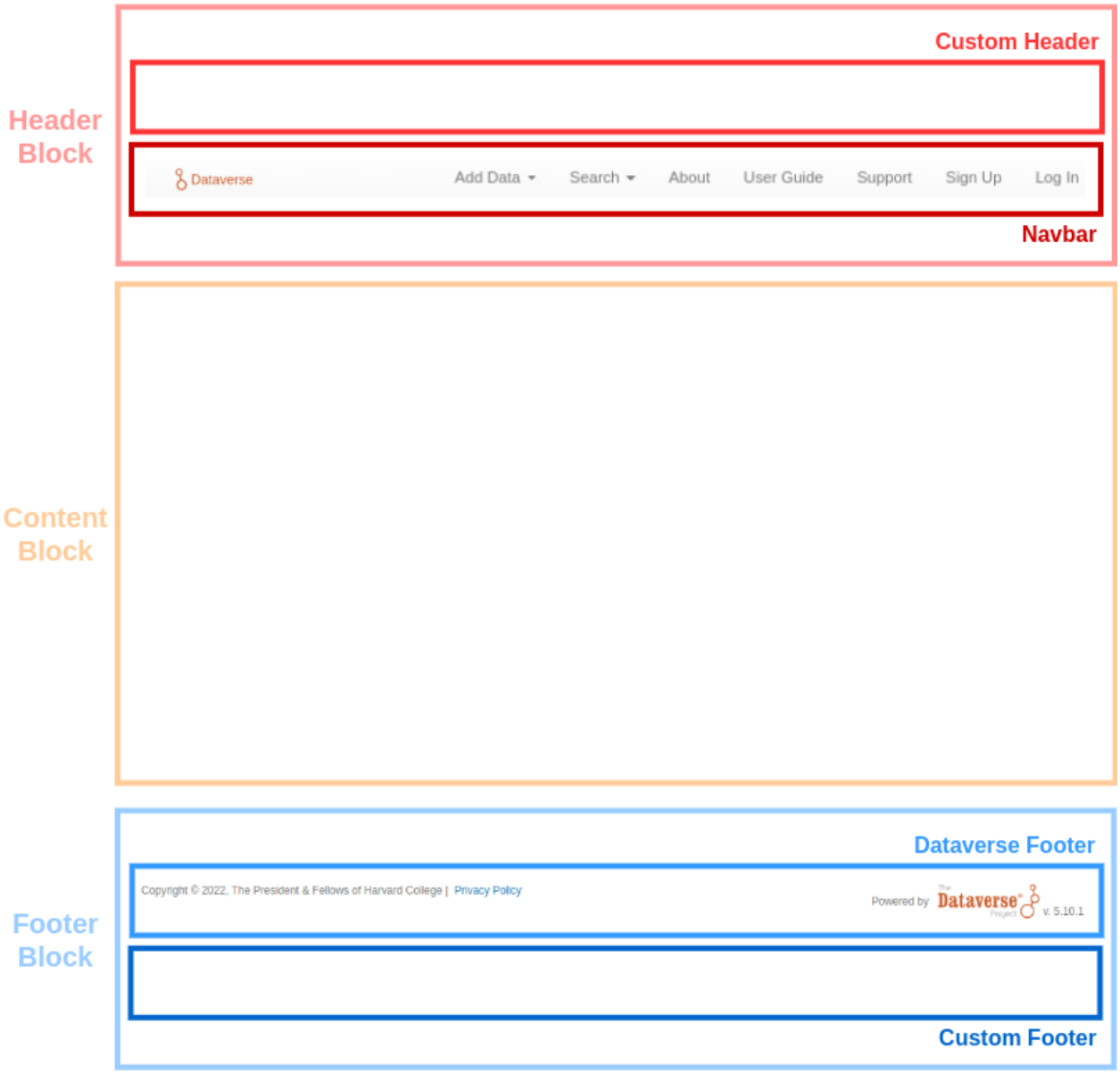
Downloadable sample HTML and CSS files are provided below which you can edit as you see fit. It's up to you to create a directory in which to store these files, such as `/var/www/dataverse` in the examples below.

Additional samples from community member Sherry Lake are available at https://github.com/shlake/LibraDataHomepage.

Parts of a Dataverse Installation Webpage

Before reading about the available customization options, you might want to familiarize yourself with the parts of a Dataverse installation webpage.

The image below indicates that the page layout consists of three main blocks: a header block, a content block, and a footer block:



Installation Name/Brand Name

It's common for a Dataverse installation to have some sort of installation name or brand name like "HeiDATA", "Libra Data", or "MELDATA".

The installation name appears in various places such as notifications, support links, and metadata exports.

Out of the box, the installation name comes from the name of root Dataverse collection ("Root" by default). You can simply change the name of this collection to set the installation name you want.

Alternatively, you can have independent names for the root Dataverse collection and the installation name by having the installation name come from the *:InstallationName* setting.

Note that you can use *dataverse.mail.system-email* to control the name that appears in the "from" address of email messages sent by a Dataverse installation. This overrides the name of the root Dataverse collection and *:InstallationName*.

If you have an image for your installation name, you can use it as the "Custom Navbar Logo", described below.

Header Block

Within the header block, you have a navbar (which will always be displayed) and you may insert a custom header that will be displayed above the navbar.

Navbar

The navbar is the component displayed by default on the header block and will be present on every Dataverse webpage.

The navbar encompasses several configurable settings (described below) that manage user interaction with a Dataverse installation.

Custom Navbar Logo

The Dataverse Software allows you to replace the default Dataverse Project icon and name branding in the navbar with your own custom logo. Note that this logo is separate from the logo used in the theme of the root Dataverse collection (see *Theme*).

The custom logo image file is expected to be small enough to fit comfortably in the navbar, no more than 50 pixels in height and 160 pixels in width. Create a `navbar` directory in your Payara `logos` directory and place your custom logo there. By default, your logo image file will be located at `/usr/local/payara7/glassfish/domains/domain1/docroot/logos/navbar/logo.png`.

Given this location for the custom logo image file, run this curl command to add it to your settings:

```
curl -X PUT -d '/logos/navbar/logo.png' http://localhost:8080/api/admin/settings/
:LogoCustomizationFile
```

To revert to the default configuration and have the Dataverse Project icon be displayed, run the following command:

```
curl -X DELETE http://localhost:8080/api/admin/settings/:LogoCustomizationFile
```

About URL

Refer to *:NavbarAboutUrl* for setting a fully-qualified URL which will be used for the "About" link in the navbar.

User Guide URL

Refer to *:NavbarGuidesUrl*, *:GuidesBaseUrl*, and *:GuidesVersion* for setting a fully-qualified URL which will be used for the "User Guide" link in the navbar.

Support URL

Refer to *:NavbarSupportUrl* for setting to a fully-qualified URL which will be used for the "Support" link in the navbar.

Sign Up

Refer to *:SignUpUrl* and *:AllowSignUp* for setting a relative path URL to which users will be sent for sign up and for controlling the ability for creating local user accounts.

Custom Header

As a starting point you can download `custom-header.html` and place it at `/var/www/dataverse/branding/custom-header.html`.

Given this location for the custom header HTML file, run this curl command to add it to your settings:

```
curl -X PUT -d '/var/www/dataverse/branding/custom-header.html' http://localhost:8080/
api/admin/settings/:HeaderCustomizationFile
```

If you have enabled a custom header or navbar logo, you might prefer to disable the theme of the root dataverse. You can do so by setting `:DisableRootDataverseTheme` to `true` like this:

```
curl -X PUT -d 'true' http://localhost:8080/api/admin/settings/:DisableRootDataverseTheme
```

Please note: Disabling the display of the root Dataverse collection theme also disables your ability to edit it. Remember that Dataverse collection owners can set their Dataverse collections to "inherit theme" from the root. Those Dataverse collections will continue to inherit the root Dataverse collection theme (even though it no longer displays on the root). If you would like to edit the root Dataverse collection theme in the future, you will have to re-enable it first.

Content Block

As shown in the image under *Parts of a Dataverse Installation Webpage*, the content block is the area below the header and above the footer.

By default, when you view the homepage of a Dataverse installation, the content block shows the root Dataverse collection. This page contains the data available in the Dataverse installation (e.g. dataverses and datasets) and the functionalities that allow the user to interact with the platform (e.g. search, create/edit data and metadata, etc.).

Rather than showing the root Dataverse collection on the homepage, the content block can show a custom homepage instead. Read on for details.

Custom Homepage

When you configure a custom homepage, it **replaces** the root Dataverse collection in the content block, serving as a welcome page. This allows for complete control over the look and feel of the content block for your installation's homepage.

As a starting point, download `custom-homepage.html` and place it at `/var/www/dataverse/branding/custom-homepage.html`.

Given this location for the custom homepage HTML file, run this curl command to add it to your settings:

```
curl -X PUT -d '/var/www/dataverse/branding/custom-homepage.html' http://localhost:8080/
api/admin/settings/:HomePageCustomizationFile
```

Note that the `custom-homepage.html` file provided has multiple elements that assume your root Dataverse collection still has an alias of "root". While you were branding your root Dataverse collection, you may have changed the alias to "harvard" or "librascholar" or whatever and you should adjust the custom homepage code as needed.

Note: If you prefer to start with less of a blank slate, you can review the custom homepage used by the Harvard Dataverse Repository, which includes branding messaging, action buttons, search input, subject links, and recent dataset links.

This page was built to utilize the *Metrics API* to deliver dynamic content to the page via Javascript. The files can be found at https://github.com/IQSS/dataverse.harvard.edu

If you decide you'd like to remove this setting, use the following curl command:

```
curl -X DELETE http://localhost:8080/api/admin/settings/:HomePageCustomizationFile
```

Footer Block

Within the footer block you have the default footer section (which will always be displayed) and you can insert a custom footer that will be displayed below the default footer.

Default Footer

The default footer is the component displayed by default on the footer block and will be present on every Dataverse webpage. Its configuration options are described below.

Footer Copyright

Refer to *:FooterCopyright* to add customized text to the Copyright section of the default Dataverse footer

Custom Footer

As mentioned above, the custom footer appears below the default footer.

As a starting point, download `custom-footer.html` and place it at `/var/www/dataverse/branding/custom-footer.html`.

Given this location for the custom footer HTML file, run this curl command to add it to your settings:

```
curl -X PUT -d '/var/www/dataverse/branding/custom-footer.html' http://localhost:8080/
api/admin/settings/:FooterCustomizationFile
```

Custom Stylesheet

You can style your custom homepage, footer, and header content with a custom CSS file. With advanced CSS know-how, you can achieve custom branding and page layouts by utilizing `position`, `padding` or `margin` properties.

As a starting point, download `custom-stylesheet.css` and place it at `/var/www/dataverse/branding/custom-stylesheet.css`.

Given this location for the custom CSS file, run this curl command to add it to your settings:

```
curl -X PUT -d '/var/www/dataverse/branding/custom-stylesheet.css' http://localhost:8080/
api/admin/settings/:StyleCustomizationFile
```

6.5.12 Internationalization

The Dataverse Software is being translated into multiple languages by the Dataverse Project Community! Please see below for how to help with this effort!

Adding Multiple Languages to the Dropdown in the Header

The presence of the *:Languages* database setting adds a dropdown in the header for multiple languages. For example to add English and French to the dropdown:

```
curl http://localhost:8080/api/admin/settings/:Languages -X PUT -d '[{"locale":"en",
"title":"English"},{"locale":"fr","title":"Français"}]'
```

When a user selects one of the available choices, the Dataverse user interfaces will be translated into that language (assuming you also configure the "lang" directory and populate it with translations as described below).

Allowing the Language Used for Dataset Metadata to be Specified

Since dataset metadata can only be entered in one language, and administrators may wish to limit which languages metadata can be entered in, Dataverse also offers a separate setting defining allowed metadata languages. The presence of the *:MetadataLanguages* database setting identifies the available options (which can be different from those in the :Languages setting above, with fewer or more options).

Dataverse collection admins can select from these options to indicate which language should be used for new Datasets created with that specific collection. If they do not, users will be asked when creating a dataset to select the language they want to use when entering metadata. Similarly, when this setting is defined, Datasets created/imported/migrated are required to specify a metadataLanguage compatible with the collection's requirement.

When creating or editing a dataset, users will be asked to enter the metadata in that language. The metadata language selected will also be shown when dataset metadata is viewed and will be included in metadata exports (as appropriate for each format) for published datasets:

```
curl http://localhost:8080/api/admin/settings/:MetadataLanguages -X PUT -d
'[{"locale":"en","title":"English"},{"locale":"fr","title":"Français"}]'
```

Note that metadata selected from Controlled Vocabularies will also display in the metadata language of the dataset, but only if translations have been configured, i.e. you configure the "lang" directory and populate it with translations as described below). In metadata export files, controlled vocabulary values will be included in the Dataverse installations default language and in the metadata language of the dataset (if specified).

Configuring the "lang" Directory

Translations for the Dataverse Software are stored in "properties" files in a directory on disk (e.g. `/home/dataverse/langBundles`) that you specify with the *dataverse.lang.directory* `dataverse.lang.directory` JVM option, like this:

```
./asadmin create-jvm-options '-Ddataverse.lang.directory=/home/dataverse/langBundles'
```

Go ahead and create the directory you specified.

```
mkdir /home/dataverse/langBundles
```

Creating a languages.zip File

The Dataverse Software provides and API endpoint for adding languages using a zip file.

First, clone the "dataverse-language-packs" git repo.

```
git clone https://github.com/GlobalDataverseCommunityConsortium/dataverse-language-packs.
git
```

Take a look at https://github.com/GlobalDataverseCommunityConsortium/dataverse-language-packs/branches to see if the version of the Dataverse Software you're running has translations.

Change to the directory for the git repo you just cloned.

```
cd dataverse-language-packs
```

Switch (`git checkout`) to the branch based on the Dataverse Software version you are running. The branch "dataverse-v4.13" is used in the example below.

```
export BRANCH_NAME=dataverse-v4.13
```

```
git checkout $BRANCH_NAME
```

Create a "languages" directory in "/tmp".

```
mkdir /tmp/languages
```

Copy the properties files into the "languages" directory

```
cp -R en_US/*.properties /tmp/languages
```

```
cp -R fr_CA/*.properties /tmp/languages
```

Create the zip file

```
cd /tmp/languages
```

```
zip languages.zip *.properties
```

Load the languages.zip file into your Dataverse Installation

Now that you have a "languages.zip" file, you can load it into your Dataverse installation with the command below.

```
curl http://localhost:8080/api/admin/datasetfield/loadpropertyfiles -X POST --upload-file
/tmp/languages/languages.zip -H "Content-Type:  application/zip"
```

Stop and start Payara and then click on the languages using the drop down in the header to try them out.

How to Help Translate the Dataverse Software Into Your Language

Please join the dataverse-internationalization-wg mailing list and contribute to https://github.com/GlobalDataverseCommunityConsortium/dataverse-language-packs to help translate the Dataverse Software into various languages!

Some external tools are also ready to be translated, especially if they are using the `{localeCode}` reserved word in their tool manifest. For details, see the *Building External Tools* section of the API Guide.

Tools for Translators

The list below depicts a set of tools that can be used to ease the amount of work necessary for translating the Dataverse software by facilitating this collaborative effort and enabling the reuse of previous work:

- Weblate for the Dataverse Software, made available in the scope of the SSHOC project.
- easyTranslationHelper, a tool developed by University of Aveiro.
- Dataverse General User Interface Translation Guide for Weblate, a guide produced as part of the SSHOC Dataverse Translation event.

6.5.13 Web Analytics Code

Your analytics code can be added to your Dataverse installation in a similar fashion to how you brand it, by adding a custom HTML file containing the analytics code snippet and adding the file location to your settings.

Popular analytics providers Google Analytics (https://www.google.com/analytics/) and Matomo (formerly "Piwik"; https://matomo.org/) have been set up to work with the Dataverse Software. Use the documentation they provide to add the analytics code to your custom HTML file. This allows for more control of your analytics, making it easier to customize what you prefer to track.

Create your own `analytics-code.html` file using the analytics code snippet provided by Google or Matomo and place it somewhere on the server, outside the application deployment directory; for example: `/var/www/dataverse/branding/analytics-code.html`. Here is an *example* of what your HTML file will look like:

```
<!-- Global Site Tag (gtag.js) - Google Analytics -->
<script async="async" src="https://www.googletagmanager.com/gtag/js?id=YOUR-ACCOUNT-CODE
↪"></script>
<script>
    //<![CDATA[
    window.dataLayer = window.dataLayer || [];
    function gtag(){dataLayer.push(arguments);}
    gtag('js', new Date());

    gtag('config', 'YOUR-ACCOUNT-CODE');
    //]]>
</script>
```

IMPORTANT: Note the "async" attribute in the first script line above. In the documentation provided by Google, its value is left blank (as in `<script async src="...">`). It must be set as in the example above (`<script async="async" src="...">`), otherwise it may cause problems with some browsers.

Once you have created the analytics file, run this curl command to add it to your settings (using the same file location as in the example above):

```
curl -X PUT -d '/var/www/dataverse/branding/analytics-code.html' http://localhost:8080/
api/admin/settings/:WebAnalyticsCode
```

Tracking Button Clicks

The basic analytics configuration above tracks page navigation. However, it does not capture potentially interesting events, such as those from users clicking buttons on pages, that do not result in a new page opening. In a Dataverse installation, these events include file downloads, requesting access to restricted data, exporting metadata, social media sharing, requesting citation text, launching external tools, contacting authors, and launching computations.

Both Google and Matomo provide the optional capability to track such events and the Dataverse Software has added CSS style classes (btn-compute, btn-contact, btn-download, btn-explore, btn-export, btn-preview, btn-request, btn-share, and downloadCitation) to it's HTML to facilitate it.

For Google Analytics, the example script at `analytics-code.html` will track both page hits and events within your Dataverse installation. You would use this file in the same way as the shorter example above, putting it somewhere outside your deployment directory, replacing `YOUR ACCOUNT CODE` with your actual code and setting :WebAnalyticsCode to reference it.

Once this script is running, you can look in the Google Analytics console (Realtime/Events or Behavior/Events) and view events by type and/or the Dataset or File the event involves.

Adding Cookie Consent (for GDPR, etc.)

Cookie consent may be required on websites that use analytics tracking codes to comply with privacy regulations, such as the GDPR, which mandate informing users about the collection and use of their data, thereby giving them the option to accept or reject cookies for enhanced privacy and control over their personal information.

Members of the Dataverse community have used the CookieConsent (https://cookieconsent.orestbida.com) JavaScript library for this purpose. Configuration advice is below.

To allow users to opt out of the use of Google Analytics tracking you can do the following:

1. Assuming you already have `analytics-code.html` added and configured, update the Google Analytics scripts by adding `type`, `data-category`, and `data-service` arguments like this:

```
<!-- Global Site Tag (gtag.js) - Google Analytics -->
<script
    async="async"
    src="https://www.googletagmanager.com/gtag/js?id=YOUR-ACCOUNT-CODE"
    type="text/plain"
    data-category="analytics"
    data-service="Google Analytics"></script>
<script
    type="text/plain"
    data-category="analytics"
    data-service="Google Analytics">
    //<![CDATA[
    window.dataLayer = window.dataLayer || [];
    function gtag(){dataLayer.push(arguments);}
    gtag('js', new Date());

    gtag('config', 'YOUR-ACCOUNT-CODE');
    //]]>
</script>
```

2. Add to `analytics-code.html`:

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/gh/orestbida/cookieconsent@v3.0.0/
dist/cookieconsent.css">
```

3. Go to https://playground.cookieconsent.orestbida.com to configure, download and copy contents of `cookieconsent-config.js` to `analytics-code.html`. It should look something like this:

```
<script type="module">
  import 'https://cdn.jsdelivr.net/gh/orestbida/cookieconsent@v3.0.0/dist/cookieconsent.
↪umd.js';

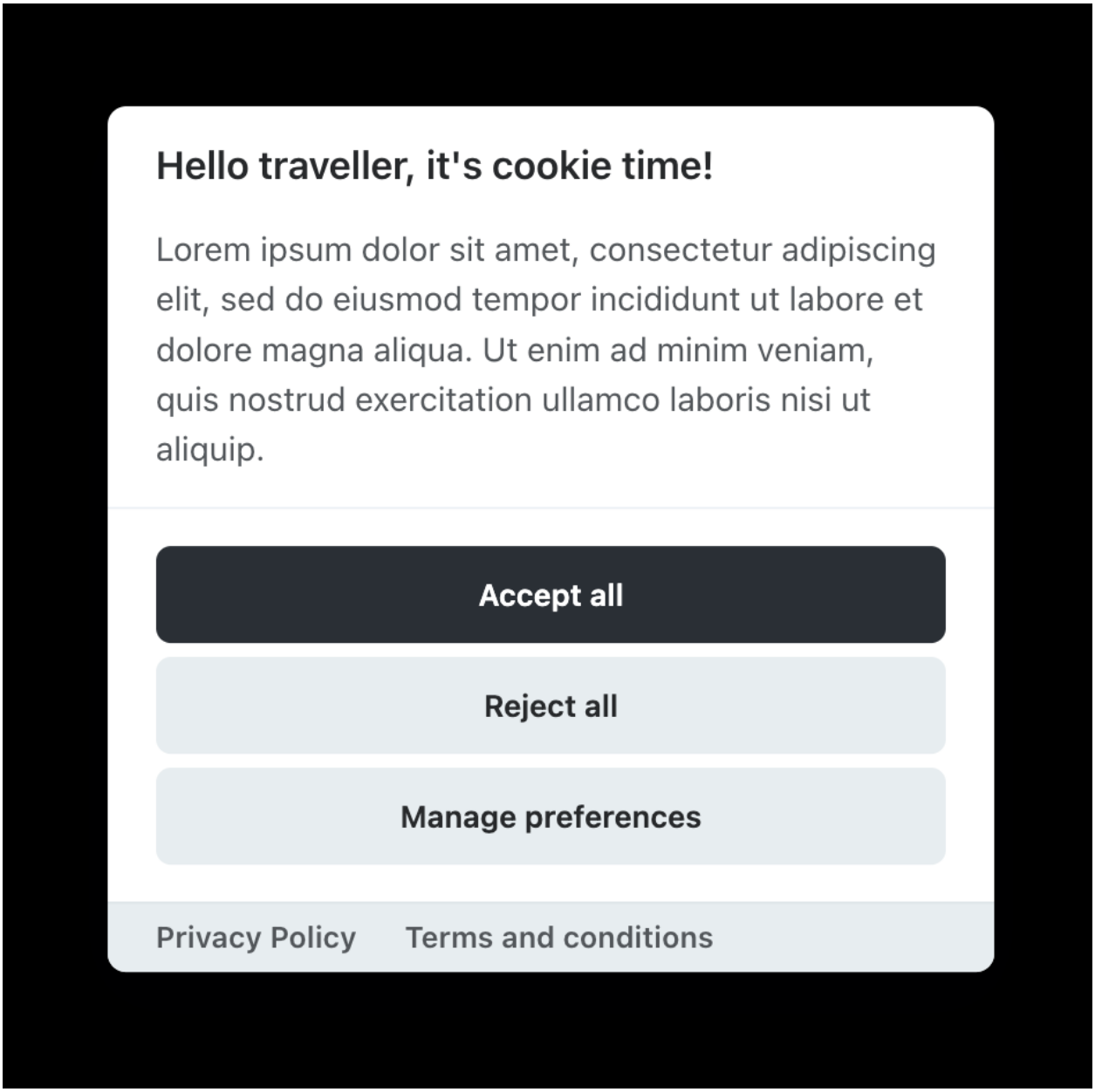
  CookieConsent.run({
    guiOptions: {
      consentModal: {
        layout: "box",
        position: "middle center",
        equalWeightButtons: false,
        flipButtons: false
      },
      preferencesModal: {
        layout: "box",
        position: "right",
        equalWeightButtons: true,
        flipButtons: false
      }
    },
    categories: {
      necessary: {
        readOnly: true
      },
      analytics: {},
    },
    language: {
      default: "en",
      autoDetect: "browser",
      translations: {
        en: {
          consentModal: {
            title: "Hello traveller, it's cookie time!",
            description: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed␣
↪do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
↪ quis nostrud exercitation ullamco laboris nisi ut aliquip.",
            acceptAllBtn: "Accept all",
            acceptNecessaryBtn: "Reject all",
            showPreferencesBtn: "Manage preferences",
            footer: "<a href=\"#link\">Privacy Policy</a>\n<a href=\"#link\">Terms and␣
↪conditions</a>"
          },
          preferencesModal: {
            title: "Consent Preferences Center",
            acceptAllBtn: "Accept all",
            acceptNecessaryBtn: "Reject all",
            savePreferencesBtn: "Save preferences",
            closeIconLabel: "Close modal",
            serviceCounterLabel: "Service|Services",
            sections: [
              {
```

(continues on next page)

(continued from previous page)

```
                title: "Cookie Usage",
                description: "Lorem ipsum dolor sit amet, consectetur adipiscing elit,␣
↪sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim␣
↪veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo␣
↪consequat."
              },
              {
                title: "Strictly Necessary Cookies <span class=\"pm__badge\">Always␣
↪Enabled</span>",
                description: "Lorem ipsum dolor sit amet, consectetur adipiscing elit,␣
↪sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim␣
↪veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo␣
↪consequat.",
                linkedCategory: "necessary"
              },
              {
                title: "Analytics Cookies",
                description: "Lorem ipsum dolor sit amet, consectetur adipiscing elit,␣
↪sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim␣
↪veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo␣
↪consequat.",
                linkedCategory: "analytics"
              },
              {
                title: "More information",
                description: "For any query in relation to my policy on cookies and your␣
↪choices, please <a class=\"cc__link\" href=\"#yourdomain.com\">contact me</a>."
              }
            ]
          }
        }
      }
    },
    disablePageInteraction: true
  });
</script>
```

After restarting or reloading Dataverse the cookie consent popup should appear, looking something like this:



If you change the cookie consent config in `CookieConsent.run()` and want to test your changes, you should remove the cookie called `cc_cookie` in your browser and reload the Dataverse page to have the popup appear again. To remove cookies use Application > Cookies in the Chrome/Edge dev tool, and Storage > Cookies in Firefox and Safari.

6.5.14 Configuring Licenses

On a new Dataverse installation, users may select from the following licenses or terms:

- CC0 1.0 (default)
- CC BY 4.0
- Custom Dataset Terms

(Note that existing Dataverse installations which are upgraded from 5.9 or previous will only offer CC0 1.0, added automatically during the upgrade to version 5.10.)

If the Dataverse Installation supports multiple languages, the license name/description translations should be added to the `License` properties files. (See *Internationalization* for more on properties files and internationalization in general.) To create the key, the license name has to be converted to lowercase, replace space with underscore.

Example:

```
license.cc0_1.0.description=Creative Commons CC0 1.0 Universal Public Domain Dedication.
license.cc0_1.0.name=CC0 1.0
```

You have a lot of control over which licenses and terms are available. You can remove licenses and add new ones. You can decide which license is the default. You can remove "Custom Dataset Terms" as a option. You can remove all licenses and make "Custom Dataset Terms" the only option.

Before making changes, you are encouraged to read the *Terms* section of the User Guide about why CC0 is the default and what the "Custom Dataset Terms" option allows.

Setting the Default License

The default license can be set with a curl command as explained in the API Guide under *Manage Available Standard License Terms*.

Note that "Custom Dataset Terms" is not a license and cannot be set to be the default.

Adding Licenses

Licenses are added with curl using JSON file as explained in the API Guide under *Manage Available Standard License Terms*.

Adding Creative Common Licenses

JSON files for Creative Commons licenses are provided below. Note that a new installation of Dataverse already includes CC0 and CC BY.

- `licenseCC0-1.0.json`
- `licenseCC-BY-4.0.json`
- `licenseCC-BY-SA-4.0.json`
- `licenseCC-BY-NC-4.0.json`
- `licenseCC-BY-NC-SA-4.0.json`
- `licenseCC-BY-ND-4.0.json`
- `licenseCC-BY-NC-ND-4.0.json`

Adding Open Data Commons Licenses

JSON files for Open Data Commons licenses are provided below.

- `licenseODbL-1.0.json`
- `licenseODC-By-1.0.json`
- `licensePDDL-1.0.json`

Adding Software Licenses

JSON files for software licenses are provided below.

- `licenseApache-2.0.json`
- `licenseMIT.json`
- `licenseEUPL-1.2.json`

Adding Country-Specific Licenses

- `licenseEtalab-2.0.json` used in France (Etalab Open License 2.0, CC-BY 2.0 compliant).
- `licenseOGL-UK-3.0.json`

Contributing to the Collection of Standard Licenses Above

If you do not find the license JSON you need above, you are encouraged to contribute it to this documentation. Following the Dataverse 6.9 release, we have standardized on the following procedure:

- Look for the license at https://spdx.org/licenses/ and https://github.com/datacite/bracco/blob/main/app/spdx.js.
- `cd scripts/api/data/licenses`.
- Copy an existing license as a starting point.
- Name your file using the SPDX identifier. For example, if the identifier is `Apache-2.0`, you should name your file `licenseApache-2.0.json`.
- For the `name` field, use the "short identifier" from the SPDX landing page (e.g. `Apache-2.0`).
- For the `shortDescription` field, use the "full name" from the SPDX landing page (e.g. `Apache License 2.0`) followed by a period (full-stop) (e.g. `Apache License 2.0.`).
- For the `uri` field, use the same resource that DataCite uses, which is often the same as the first "Other web pages for this license" on the SPDX page for the license. Look at the `seeAlso` array for the license at https://github.com/datacite/bracco/blob/main/app/spdx.js to be sure. When these differ, or there are other concerns about the URI DataCite uses, please reach out to the community to see if a consensus can be reached. See *Getting Help*.
- For the `active` field, put `true`.
- For the `sortOrder` field, put the next sequential number after checking previous files with `grep sortOrder scripts/api/data/licenses/*`.
- For the `rightsIdentifier` field, use the "short identifier" from the SPDX landing page (e.g. `Apache-2.0`).
- For the `rightsIdentifierScheme` field, use "SPDX".

- For the `schemeUri` field, use "https://spdx.org/licenses/".
- For the `languageCode` field, use "en".
- For all of the fields above, resist the urge to change the spelling of words like license/licence, center/centre, etc. SPDX is the upstream authority, and they have the following varietal word spelling policy: "The words in each line of the text file available at https://spdx.org/licenses/equivalentwords.txt are considered equivalent and interchangeable."

In the past, licenses have been added that do not adhere perfectly with the procedure above. Here are known inconsistencies:

- The `name` for the CC licenses don't have a dash as their SPDX short identifiers do (e.g. CC-BY-4.0, CC-BY-NC-4.0, CC-BY-NC-ND-4.0, CC-BY-NC-SA-4.0, CC-BY-ND-4.0, CC-BY-SA-4.0, CC0-1.0). For example, the `name` for the CC0 license is `CC0 1.0` (no dash) rather than `CC0-1.0` (with a dash). We are keeping the existing names without dashes for backward compatibility.
- The `uri` for Creative Commons licenses comes from the Creative Commons website rather than SPDX or DataCite. As with `name`, we are keeping `uri` the as-is for these licenses for backward compatibility. For more on our attempts to standardize license configuration, see https://github.com/IQSS/dataverse/issues/8512 and https://github.com/IQSS/dataverse/pull/11522.
- The `uri` for Etalab is https://spdx.org/licenses/etalab-2.0 rather than a link listed in SPDX or DataCite.
- The `shortDescription` doesn't have a trailing period for Apache-2.0, Etalab, and MIT.

Adding Custom Licenses

If you are interested in adding a custom license, you will need to create your own JSON file as explained in see *Standardizing Custom Licenses*.

Removing Licenses

Licenses can be removed with a curl command as explained in the API Guide under *Manage Available Standard License Terms*.

Disabling Custom Dataset Terms

See *:AllowCustomTermsOfUse* for how to disable the "Custom Dataset Terms" option.

6.5.15 Sorting licenses

The default order of licenses in the dropdown in the user interface is as follows:

- The default license is shown first
- Followed by the remaining installed licenses in the order of installation
- The custom license is at the end

Only the order of the installed licenses can be changed with the API calls. The default license always remains first and the custom license last.

The order of licenses can be changed by setting the `sortOrder` property of a license. For the purpose of making sorting easier and to allow grouping of the licenses, `sortOrder` property does not have to be unique. Licenses with the same `sortOrder` are sorted by their ID, i.e., first by the sortOrder, then by the ID. Nevertheless, you can set a unique `sortOrder` for every license in order to sort them fully manually.

The `sortOrder` is an whole number and is used to sort licenses in ascending fashion.

Changing the sorting order of a license specified by the license `$ID` is done by superusers using the following API call:

```
export SORT_ORDER=100
curl -X PUT -H 'Content-Type: application/json' -H X-Dataverse-key:$API_TOKEN $SERVER_
↪URL/api/licenses/$ID/:sortOrder/$SORT_ORDER
```

6.5.16 BagIt File Handler

BagIt file handler detects and transforms zip files with a BagIt package format into Dataverse DataFiles. The system validates the checksums of the files in the package payload as described in the first manifest file with a hash algorithm that we support. Take a look at BagChecksumType class for the list of the currently supported hash algorithms.

The checksum validation uses a thread pool to improve performance. This thread pool can be adjusted to your Dataverse installation requirements.

BagIt file handler configuration settings:

- *:BagItHandlerEnabled*
- *:BagValidatorJobPoolSize*
- *:BagValidatorMaxErrors*
- *:BagValidatorJobWaitInterval*

6.5.17 BagIt Export

Your Dataverse installation may be configured to submit a copy of published Datasets, packaged as Research Data Alliance conformant zipped BagIt archival Bags (sometimes called BagPacks) to one of several supported storage services. Supported services include Chronopolis via DuraCloud, Google's Cloud, and any service that can provide an S3 interface or handle files transferred to a folder on the local filesystem.

These archival Bags include all of the files and metadata in a given dataset version and are sufficient to recreate the dataset, e.g. in a new Dataverse instance, or potentially in another RDA-conformant repository. The DVUploader includes functionality to recreate a Dataset from an archival Bag produced by Dataverse. (Note that this functionality is distinct from the *BagIt File Handler* upload files to an existing Dataset via the Dataverse user interface.)

The Dataverse Software offers an internal archive workflow which may be configured as a PostPublication workflow via an admin API call to manually submit previously published Datasets and prior versions to a configured archive such as Chronopolis. The workflow creates a JSON-LD serialized OAI-ORE map file, which is also available as a metadata export format in the Dataverse Software web interface.

The size of the zipped archival Bag can be limited, and files that don't fit within that limit can either be transferred separately (placed so that they are correctly positioned according to the BagIt specification when the zipped bag in unzipped in place) or just referenced for later download (using the BagIt concept of a 'holey' bag with a list of files in a `fetch.txt` file) can now be configured for all archivers. These settings allow for managing large datasets by excluding files over a certain size or total data size, which can be useful for archivers with size limitations or to reduce transfer times. See the *dataverse.bagit.zip.max-file-size*, *dataverse.bagit.zip.max-data-size*, and *dataverse.bagit.zip.holey* JVM options for more details.

At present, archiving classes include the DuraCloudSubmitToArchiveCommand, LocalSubmitToArchiveCommand, GoogleCloudSubmitToArchive, and S3SubmitToArchiveCommand , which all extend the AbstractSubmitToArchiveCommand and use the configurable mechanisms discussed below. (A DRSSubmitToArchiveCommand, which works with Harvard's DRS also exists and, while specific to DRS, is a useful example of how Archivers can

support single-version-only semantics and support archiving only from specified collections (with collection specific parameters)).

All current options support the *Get the Archival Status of a Dataset By Version* calls and the same status is available in the dataset page version table (for contributors/those who could view the unpublished dataset, with more detail available to superusers).

Two settings that can be used with all current Archivers are:

- :BagGeneratorThreads - the number of threads to use when adding data files to the zipped bag. The default is 2. Values of 4 or more may increase performance on larger machines but may cause problems if file access is throttled
- :ArchiveOnlyIfEarlierVersionsAreArchived - when true, requires dataset versions to be archived in order by confirming that all prior versions have been successfully archived before allowing a new version to be archived. Default is false

These must be included in the :ArchiverSettings for the Archiver to work

Archival Bags are created per dataset version. By default, if a version is republished (via the superuser-only 'Update Current Version' publication option in the UI/API), a new archival bag is not created for the version. If the archiver used is capable of deleting existing bags (Google, S3, and File Archivers) superusers can trigger a manual update of the archival bag, and, if the *dataverse.bagit.archive-on-version-update* flag is set to true, this will be done automatically when 'Update Current Version' is used.

Duracloud Configuration

The current Chronopolis implementation generates the archival Bag and submits it to the archive's DuraCloud interface. The step to make a 'snapshot' of the space containing the archival Bag (and verify it's successful submission) are actions a curator must take in the DuraCloud interface.

The minimal configuration to support archiver integration involves adding a minimum of two Dataverse Software settings. Individual archivers may require additional settings and/or Payara jvm options and micro-profile settings. The example instructions here are specific to the DuraCloud Archiver:

:ArchiverClassName - the fully qualified class to be used for archiving. For example:

```
curl http://localhost:8080/api/admin/settings/:ArchiverClassName -X PUT -d "edu.harvard.
iq.dataverse.engine.command.impl.DuraCloudSubmitToArchiveCommand"
```

:ArchiverSettings - the archiver class can access required settings including existing Dataverse installation settings and dynamically defined ones specific to the class. This setting is a comma-separated list of those settings. For example:

```
curl http://localhost:8080/api/admin/settings/:ArchiverSettings -X PUT -d
":DuraCloudHost, :DuraCloudPort, :DuraCloudContext, :BagGeneratorThreads"
```

The DuraCloud archiver defines three custom settings, one of which is required (the others have defaults):

:DuraCloudHost - the URL for your organization's Duracloud site. For example:

```
curl http://localhost:8080/api/admin/settings/:DuraCloudHost -X PUT -d "qdr.duracloud.
org"
```

:DuraCloudPort and :DuraCloudContext are also defined if you are not using the defaults ("443" and "duracloud" respectively). (Note: these settings are only in effect if they are listed in the :ArchiverSettings. Otherwise, they will not be passed to the DuraCloud Archiver class.)

It also can use one setting that is common to all Archivers: :BagGeneratorThreads

```
curl http://localhost:8080/api/admin/settings/:BagGeneratorThreads -X PUT -d '8'
```

By default, the Bag generator zips two datafiles at a time when creating the archival Bag. This setting can be used to lower that to 1, i.e. to decrease system load, or to increase it, e.g. to 4 or 8, to speed processing of many small files.

Archivers may require JVM options as well. For the Chronopolis archiver, the username and password associated with your organization's Chronopolis/DuraCloud account should be configured in Payara:

```
./asadmin create-jvm-options '-Dduracloud.username=YOUR_USERNAME_HERE'
```

```
./asadmin create-jvm-options '-Dduracloud.password=YOUR_PASSWORD_HERE'
```

Local Path Configuration

ArchiverClassName - the fully qualified class to be used for archiving. For example:

```
curl -X PUT -d "edu.harvard.iq.dataverse.engine.command.impl.LocalSubmitToArchiveCommand"
http://localhost:8080/api/admin/settings/:ArchiverClassName
```

:BagItLocalPath - the path to where you want to store the archival Bags. For example:

```
curl -X PUT -d /home/path/to/storage http://localhost:8080/api/admin/settings/
:BagItLocalPath
```

:ArchiverSettings - the archiver class can access required settings including existing Dataverse installation settings and dynamically defined ones specific to the class. This setting is a comma-separated list of those settings. For example:

```
curl http://localhost:8080/api/admin/settings/:ArchiverSettings -X PUT -d
":BagItLocalPath, :BagGeneratorThreads"
```

:BagItLocalPath is the file path that you've set in :ArchiverSettings. See the DuraCloud Configuration section for a description of :BagGeneratorThreads.

Google Cloud Configuration

The Google Cloud Archiver can send Dataverse Archival Bags to a bucket in Google's cloud, including those in the 'Coldline' storage class (cheaper, with slower access)

```
curl http://localhost:8080/api/admin/settings/:ArchiverClassName -X PUT -d "edu.harvard.
iq.dataverse.engine.command.impl.GoogleCloudSubmitToArchiveCommand"
```

```
curl http://localhost:8080/api/admin/settings/:ArchiverSettings -X PUT -d
":GoogleCloudBucket, :GoogleCloudProject, :BagGeneratorThreads"
```

The Google Cloud Archiver defines two custom settings, both are required. It can also use the :BagGeneratorThreads setting as described in the DuraCloud Configuration section above. The credentials for your account, in the form of a json key file, must also be obtained and stored locally (see below):

In order to use the Google Cloud Archiver, you must have a Google account. You will need to create a project and bucket within that account and provide those values in the settings:

:GoogleCloudBucket - the name of the bucket to use. For example:

```
curl http://localhost:8080/api/admin/settings/:GoogleCloudBucket -X PUT -d "qdr-archive"
```

:GoogleCloudProject - the name of the project managing the bucket. For example:

```
curl http://localhost:8080/api/admin/settings/:GoogleCloudProject -X PUT -d "qdr-project"
```

The Google Cloud Archiver also requires a key file that must be renamed to 'googlecloudkey.json' and placed in the directory identified by your 'dataverse.files.directory' jvm option. This file can be created in the Google Cloud Console. (One method: Navigate to your Project 'Settings'/'Service Accounts', create an account, give this account the 'Cloud Storage'/'Storage Admin' role, and once it's created, use the 'Actions' menu to 'Create Key', selecting the 'JSON' format option. Use this as the 'googlecloudkey.json' file.)

For example:

```
cp <your key file> /usr/local/payara7/glassfish/domains/domain1/files/googlecloudkey.json
```

S3 Configuration

The S3 Archiver can send Dataverse Archival Bag to a bucket at any S3 endpoint. The configuration for the S3 Archiver is independent of any S3 store that may be configured in Dataverse and may, for example, leverage colder (cheaper, slower access) storage.

```
curl http://localhost:8080/api/admin/settings/:ArchiverClassName -X PUT -d "edu.harvard.
iq.dataverse.engine.command.impl.S3SubmitToArchiveCommand"
```

```
curl http://localhost:8080/api/admin/settings/:ArchiverSettings -X PUT -d
":S3ArchiverConfig, :BagGeneratorThreads"
```

The S3 Archiver defines one custom setting, a required :S3ArchiverConfig. It can also use the :BagGeneratorThreads setting as described in the DuraCloud Configuration section above.

The credentials for your S3 account, can be stored in a profile in a standard credentials file (e.g. ~/.aws/credentials) referenced via "profile" key in the :S3ArchiverConfig setting (will default to the default entry), or can via MicroProfile settings as described for S3 stores (dataverse.s3archiver.access-key and dataverse.s3archiver.secret-key)

The :S3ArchiverConfig setting is a JSON object that must include an "s3_bucket_name" and may include additional S3-related parameters as described for S3 Stores, including "profile", "connection-pool-size","custom-endpoint-url", "custom-endpoint-region", "path-style-access", and "chunked-encoding".

:S3ArchiverConfig - minimally includes the name of the bucket to use. For example:

```
curl http://localhost:8080/api/admin/settings/:S3ArchiverConfig -X PUT -d
'{"s3_bucket_name":"archival-bucket"}'
```

:S3ArchiverConfig - example to also set the name of an S3 profile to use. For example:

```
curl http://localhost:8080/api/admin/settings/:S3ArchiverConfig -X PUT -d
'{"s3_bucket_name":"archival-bucket", "profile":"archiver"}'
```

BagIt Export API Calls

Once this configuration is complete, you, as a user with the *PublishDataset* permission, should be able to use the admin API call to manually submit a DatasetVersion for processing:

```
curl -X POST -H "X-Dataverse-key:  <key>" http://localhost:8080/api/admin/
submitDatasetVersionToArchive/{id}/{version}
```

where:

`{id}` is the DatasetId (the database id of the dataset) and

`{version}` is the friendly version number, e.g. "1.2".

or in place of the DatasetID, you can use the string `:persistentId` as the `{id}` and add the DOI/PID as a query parameter like this: `?persistentId="<DOI>"`. Here is how the full command would look:

```
curl -X POST -H "X-Dataverse-key:  <key>" http://localhost:8080/api/admin/
submitDatasetVersionToArchive/:persistentId/{version}?persistentId="<DOI>"
```

The submitDatasetVersionToArchive API (and the workflow discussed below) attempt to archive the dataset version via an archive specific method. For Chronopolis, a DuraCloud space named for the dataset (its DOI with ":" and "." replaced with "-", e.g. `doi-10-5072-fk2-tgbhlb`) is created and two files are uploaded to it: a version-specific datacite.xml metadata file and a BagIt bag containing the data and an OAI-ORE map file. (The datacite.xml file, stored

outside the Bag as well as inside, is intended to aid in discovery while the ORE map file is "complete", containing all user-entered metadata and is intended as an archival record.)

In the Chronopolis case, since the transfer from the DuraCloud front-end to archival storage in Chronopolis can take significant time, it is currently up to the admin/curator to submit a 'snap-shot' of the space within DuraCloud and to monitor its successful transfer. Once transfer is complete the space should be deleted, at which point the Dataverse Software API call can be used to submit a Bag for other versions of the same dataset. (The space is reused, so that archival copies of different dataset versions correspond to different snapshots of the same DuraCloud space.).

A batch version of this admin API call is also available:

```
curl -X POST -H "X-Dataverse-key:  <key>" 'http://localhost:8080/api/admin/
archiveAllUnarchivedDatasetVersions?listonly=true&limit=10&latestonly=true'
```

The archiveAllUnarchivedDatasetVersions call takes 3 optional configuration parameters.

- listonly=true will cause the API to list dataset versions that would be archived but will not take any action.
- limit=<n> will limit the number of dataset versions archived in one API call to <= <n>.
- latestonly=true will limit archiving to only the latest published versions of datasets instead of archiving all unarchived versions.

Note that because archiving is done asynchronously, the calls above will return OK even if the user does not have the *PublishDataset* permission on the dataset(s) involved. Failures are indicated in the log and the archivalStatus calls in the native API can be used to check the status as well.

PostPublication Workflow

To automate the submission of archival copies to an archive as part of publication, one can setup a Dataverse Software Workflow using the "archiver" workflow step - see the *Workflows* guide.

The archiver step uses the configuration information discussed above including the :ArchiverClassName setting. The workflow step definition should include the set of properties defined in :ArchiverSettings in the workflow definition.

To active this workflow, one must first install a workflow using the archiver step. A simple workflow that invokes the archiver step configured to submit to DuraCloud as its only action is included in the Dataverse Software at /scripts/api/data/workflows/internal-archiver-workflow.json.

Using the Workflow Native API (see the *Native API* guide) this workflow can be installed using:

```
curl -X POST -H 'Content-type:  application/json' --upload-file internal-archiver-workflow.
json http://localhost:8080/api/admin/workflows
```

The workflow id returned in this call (or available by doing a GET of /api/admin/workflows) can then be submitted as the default PostPublication workflow:

```
curl -X PUT -d {id} http://localhost:8080/api/admin/workflows/default/PostPublishDataset
```

Once these steps are taken, new publication requests will automatically trigger submission of an archival copy to the specified archiver, Chronopolis' DuraCloud component in this example. For Chronopolis, as when using the API, it is currently the admin's responsibility to snap-shot the DuraCloud space and monitor the result. Failure of the workflow, (e.g. if DuraCloud is unavailable, the configuration is wrong, or the space for this dataset already exists due to a prior publication action or use of the API), will create a failure message but will not affect publication itself.

Note: setting the default workflow is also available via the Settings API. See *:WorkflowsAdminIpWhitelist*, *:PrePublishDatasetWorkflowId* and *:PostPublishDatasetWorkflowId*

Configuring bag-info.txt

Out of the box, placeholder values like below will be placed in bag-info.txt:

```
Source-Organization: Dataverse Installation (<Site Url>)
Organization-Address: <Full address>
Organization-Email: <Email address>
```

To customize these values for your institution, use the following JVM options:

- *dataverse.bagit.sourceorg.name*
- *dataverse.bagit.sourceorg.address*
- *dataverse.bagit.sourceorg.email*

6.5.18 Going Live: Launching Your Production Deployment

This guide has attempted to take you from kicking the tires on the Dataverse Software to finalizing your installation before letting real users in. In theory, all this work could be done on a single server but better would be to have separate staging and production environments so that you can deploy upgrades to staging before deploying to production. This "Going Live" section is about launching your **production** environment.

Before going live with your Dataverse installation, you must take the steps above under "Securing Your Installation" and you should at least review the various configuration options listed below. An attempt has been made to put the more commonly-configured options earlier in the list.

Out of the box, the Dataverse Software attempts to block search engines from crawling your Dataverse installation so that test datasets do not appear in search results until you're ready.

Letting Search Engines Crawl Your Installation

Ensure robots.txt Is Not Blocking Search Engines

For a public production Dataverse installation, it is probably desired that search agents be able to index published pages (AKA - pages that are visible to an unauthenticated user). Polite crawlers usually respect the Robots Exclusion Standard; we have provided an example of a production robots.txt `here`).

We **strongly recommend** using the crawler rules in the sample robots.txt linked above. Note that they make the Dataverse collection and dataset pages accessible to the search engine bots; but discourage them from actually crawling the site, by following any search links - facets and such - on the Dataverse collection pages. Such crawling is very inefficient in terms of system resources, and often results in confusing search results for the end users of the search engines (for example, when partial search results are indexed as individual pages).

The recommended solution instead is to directly point the bots to the dataset and Dataverse collection pages that need to be indexed, by advertising them via an explicit sitemap (please see the next section for details on how to generate the sitemap).

You can of course modify your own robots.txt to suit your specific needs as necessary. If you don't want your datasets to be indexed at all, you can tell the bots to stay away from your site completely. But, as noted above, keep in mind that only the good, "polite" bots honor these rules! You are not really blocking anyone from accessing your site by adding a "Disallow" rule in robots.txt - it is a suggestion only. A rogue bot can and will violate it. If you are having trouble with the site being overloaded with what looks like heavy automated crawling, you may have to resort to blocking this traffic by other means - for example, via rewrite rules in Apache, or even by a Firewall.

(See the sample robots.txt file linked above for some comments on how to set up different "Allow" and "Disallow" rules for different crawler bots)

You have a couple of options for putting an updated robots.txt file into production. If you are fronting Payara with Apache as recommended above, you can place robots.txt in the root of the directory specified in your `VirtualHost` and to your Apache config a `ProxyPassMatch` line like the one below to prevent Payara from serving the version of robots.txt that is embedded in the Dataverse Software war file:

```
# don't let Payara serve its version of robots.txt
ProxyPassMatch ^/robots.txt$ !
```

For more of an explanation of `ProxyPassMatch` see the *Shibboleth* section.

If you are not fronting Payara with Apache you'll need to prevent Payara from serving the robots.txt file embedded in the war file by overwriting robots.txt after the war file has been deployed. The downside of this technique is that you will have to remember to overwrite robots.txt in the "exploded" war file each time you deploy the war file, which probably means each time you upgrade to a new version of the Dataverse Software. Furthermore, since the version of the Dataverse Software is always incrementing and the version can be part of the file path, you will need to be conscious of where on disk you need to replace the file. For example, for Dataverse Software 4.6.1 the path to robots.txt may be `/usr/local/payara7/glassfish/domains/domain1/applications/dataverse-4.6.1/robots.txt` with the version number `4.6.1` as part of the path.

Creating a Sitemap and Submitting it to Search Engines

Creating a Sitemap

Search engines have an easier time indexing content when you provide them a sitemap. Dataverse can generate a sitemap that includes URLs to all published collections and all published datasets that are not harvested or deaccessioned.

Create or update your sitemap by adding the following curl command to cron to run nightly or as you see fit:

```
curl -X POST http://localhost:8080/api/admin/sitemap
```

On a Dataverse installation with many datasets, the creation or updating of the sitemap can take a while. You can check Payara's server.log file for "BEGIN updateSiteMap" and "END updateSiteMap" lines to know when the process started and stopped and any errors in between.

For compliance with the Sitemap protocol, the generated sitemap will be a single file with 50,000 items or fewer or it will be split into multiple files.

Single Sitemap File

If you have 50,000 items or fewer, a single sitemap will be generated in the following location (unless you have customized your installation directory for Payara):

```
/usr/local/payara7/glassfish/domains/domain1/docroot/sitemap/sitemap.xml
```

Once the sitemap has been generated in the location above, it will be served at `/sitemap.xml` like this: https://demo.dataverse.org/sitemap.xml

Multiple Sitemap Files (Sitemap Index File)

According to the Sitemaps.org protocol, a sitemap file must have no more than 50,000 URLs and must be no larger than 50MiB. In this case, the protocol instructs you to create a sitemap index file called `sitemap_index.xml` (instead of `sitemap.xml`), which references multiple sitemap files named `sitemap1.xml`, `sitemap2.xml`, etc. These referenced files are also generated in the same place as other sitemap files (`domain1/docroot/sitemap`) and there will be as many files as necessary to contain the URLs of collections and datasets present in your installation, while respecting the limit of 50,000 URLs per file.

If you have over 50,000 items, a sitemap index file will be generated in the following location (unless you have customized your installation directory for Payara):

`/usr/local/payara7/glassfish/domains/domain1/docroot/sitemap/sitemap_index.xml`

Once the sitemap has been generated in the location above, it will be served at `/sitemap_index.xml` like this: https://demo.dataverse.org/sitemap_index.xml

Note that the sitemap is also available at (for example) https://demo.dataverse.org/sitemap/sitemap_index.xml and in that `sitemap` directory you will find the files it references such as `sitemap1.xml`, `sitemap2.xml`, etc.

Submitting Your Sitemap to Search Engines

Some search engines will be able to find your sitemap file at `/sitemap.xml` or `/sitemap_index.xml`, but others, **including Google**, need to be **specifically instructed** to retrieve it.

As described above, Dataverse will automatically detect whether you need to create a single sitemap file or several files and generate them for you. However, when submitting your sitemap file to Google or other search engines, you must be careful to supply the correct file name (`sitemap.xml` or `sitemap_index.xml`) depending on your situation.

One way to submit your sitemap URL to Google is by using their "Search Console" (https://search.google.com/search-console). In order to use the console, you will need to authenticate yourself as the owner of your Dataverse site. Various authentication methods are provided; but if you are already using Google Analytics, the easiest way is to use that account. Make sure you are logged in on Google with the account that has the edit permission on your Google Analytics property; go to the Search Console and enter the root URL of your Dataverse installation, then choose Google Analytics as the authentication method. Once logged in, click on "Sitemaps" in the menu on the left. Consult Google's "submit a sitemap" instructions for more information.

Putting Your Dataverse Installation on the Map at dataverse.org

Congratulations! You've gone live! It's time to announce your new data repository to the world! You are also welcome to contact support@dataverse.org to have the Dataverse Project Team add your installation to the map at http://dataverse.org . Thank you for installing the Dataverse Software!

Administration of Your Dataverse Installation

Now that you're live you'll want to review the *Admin Guide* for more information about the ongoing administration of a Dataverse installation.

Setting Up Integrations

Before going live, you might want to consider setting up integrations to make it easier for your users to deposit or explore data. See the *Integrations* section of the Admin Guide for details.

6.5.19 Comma-Separated Configuration Values

Many configuration options (both MicroProfile/JVM settings and database settings) accept comma-separated lists. For all such settings, Dataverse applies consistent, lightweight parsing:

- Whitespace immediately around commas is ignored (e.g., `GET, POST` is equivalent to `GET,POST`).
- Tokens are otherwise preserved exactly as typed. There is no quote parsing and no escape processing.
- Embedded commas within a token are not supported.

Examples include (but are not limited to):

- *dataverse.cors.origin*
- *dataverse.cors.methods*
- *dataverse.cors.headers.allow*
- *dataverse.cors.headers.expose*
- *:UploadMethods*

This behavior is implemented centrally and applies across all Dataverse settings that accept comma-separated values.

6.5.20 JVM Options

JVM stands for Java Virtual Machine and as a Java application, Payara can read JVM options when it is started. A number of JVM options are configured by the installer below is a complete list of the Dataverse Software-specific JVM options. You can inspect the configured options by running:

```
./asadmin list-jvm-options | egrep 'dataverse|doi'
```

When changing values these values with `asadmin`, you'll need to delete the old value before adding a new one, like this:

```
./asadmin delete-jvm-options "-Ddataverse.fqdn=old.example.com"
```

```
./asadmin create-jvm-options "-Ddataverse.fqdn=dataverse.example.com"
```

It's also possible to change these values by stopping Payara, editing `payara7/glassfish/domains/domain1/config/domain.xml`, and restarting Payara.

In addition, JVM options enabled for "MicroProfile Config" (see docs of any option), can be used with any supported MicroProfile Config API source to provide their values. The most notable source are environment variables; many examples are given in detail documentation of enabled options.

dataverse.fqdn

If the Dataverse installation has multiple DNS names, this option specifies the one to be used as the "official" hostname. For example, you may want to have `dataverse.example.edu`, and not the less appealing `server-123.example.edu` to appear exclusively in all the registered global identifiers, etc.

- Email confirmation links
- Password reset links
- Generating a Private URL
- PID minting
- Exporting to Schema.org format (and showing JSON-LD in HTML's <meta/> tag)
- Exporting to DDI format
- Which Dataverse installation an "external tool" should return to
- URLs embedded in SWORD API responses
- …

Usually it will follow the pattern `https://<full-qualified-domain-name>/<some-place-to-go-to>`. *Only* the FQDN part of your Dataverse installation URL can be determined by setting `dataverse.fqdn`.

Notes:

- The URL will default to using `https://` and no additional port information. If that does not suit your setup, you can define an additional option, `dataverse.siteUrl`, *explained below*, which always takes precedence.
- Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_FQDN`. Defaults to `localhost` when used with `mp.config.profile=ct`

dataverse.siteUrl

`dataverse.siteUrl` is used to configure the URL for your Dataverse installation that you plan to advertise to your users. As explained in the *installation* docs, this setting is critical for the correct operation of your installation. For example, your site URL could be https://dataverse.example.edu . That is, even though the server might also be available at uglier URLs such as https://server-123.example.edu, the site URL is the "official" URL.

That said, some environments may require using a different URL pattern to access your installation. You might need to use HTTP without "S", a non-standard port and so on. This is especially useful in development or testing environments.

You can provide any custom tailored site URL via `dataverse.siteUrl`, which always takes precedence. Example: `dataverse.siteUrl=http://localhost:8080`

If you wish to change your site URL by changing the domain configuration, you should edit your `domain.xml` directly to avoid problems with colons in commands. Find a line similar to `<jvm-options>-Ddataverse.siteUrl=https://dataverse.example.edu</jvm-options>` and change it. You can specify the protocol, host, and port number and should not include a trailing slash.

Notes:

- This setting may be used in combination with variable replacement, referencing *dataverse.fqdn* with `./asadmin create-jvm-options "\-Ddataverse.siteUrl=http\://\${dataverse.fqdn}\:8080"`
- Remember to restart Payara after editing `domain.xml`.
- Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_SITEURL`. Defaults to `http://${dataverse.fqdn}:8080` when used with `mp.config.profile=ct`

- We are absolutely aware that it's confusing to have both `dataverse.fqdn` and `dataverse.siteUrl`. https://github.com/IQSS/dataverse/issues/6636 is about resolving this confusion.

dataverse.files.directory

Providing an explicit location here makes it easier to reuse some mounted filesystem and we recommend doing so to avoid filled up disks, aid in performance, etc. This directory is used for a number of purposes:

1. `<dataverse.files.directory>/temp` after uploading, data is temporarily stored here for ingest and/or before shipping to the final storage destination.
2. `<dataverse.files.directory>/sword` a place to store uploads via the *SWORD API* before transfer to final storage location and/or ingest.
3. `<dataverse.files.directory>/googlecloudkey.json` used with *Google Cloud Configuration* for BagIt exports. This location is deprecated and might be refactored into a distinct setting in the future.
4. The experimental DCM feature for *Big Data Support* is able to trigger imports for externally uploaded files in a directory tree at `<dataverse.files.directory>/<PID Authority>/<PID Identifier>` under certain conditions. This directory may also be used by file stores for *permanent file storage*, but this is controlled by other, store-specific settings.

Notes:

- Please provide an absolute path to a directory backed by some mounted file system.
- Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_FILES_DIRECTORY`.
- Defaults to `/tmp/dataverse` in a *default installation*.
- Defaults to `${STORAGE_DIR}` using our *Dataverse container* (resolving to `/dv`).
- During startup, this directory will be checked for existence and write access. It will be created for you if missing. If it cannot be created or does not have proper write access, application deployment will fail.

dataverse.files.default-dataset-file-count-limit

Configure a limit to the maximum number of Datafiles that can be uploaded to a Dataset.

Notes:

- This is a default that can be overwritten in any Dataverse/Collection or Dataset.
- A value less than 1 will be treated as no limit set.
- Changing this value will not delete any existing files. It is only intended for preventing new files from being uploaded.
- Superusers will not be governed by this rule.

dataverse.files.uploads

Configure a folder to store the incoming file stream during uploads (before transfering to `${dataverse.files.directory}/temp`). Providing an explicit location here makes it easier to reuse some mounted filesystem. Please also see *Temporary Upload File Storage* for more details.

Notes:

- Please provide an absolute path to a directory backed by some mounted file system.
- Defaults to `${com.sun.aas.instanceRoot}/uploads` in a *default installation* (resolving to `/usr/local/payara7/glassfish/domains/domain1/uploads`).
- Defaults to `${STORAGE_DIR}/uploads` using our *Dataverse container* (resolving to `/dv/uploads`).
- Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_FILES_UPLOADS`.
- During startup, this directory will be checked for existence and write access. It will be created for you if missing. If it cannot be created or does not have proper write access, application deployment will fail.

dataverse.files.docroot

Configure a folder to store and retrieve additional materials like user uploaded collection logos, generated sitemaps, and so on. Providing an explicit location here makes it easier to reuse some mounted filesystem. See also logo customization above.

Notes:

- Defaults to `${com.sun.aas.instanceRoot}/docroot` in a *default installation* (resolves to `/usr/local/payara7/glassfish/domains/domain1/docroot`).
- Defaults to `${STORAGE_DIR}/docroot` using our *Dataverse container* (resolving to `/dv/docroot`).
- Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_FILES_DOCROOT`.
- During startup, this directory will be checked for existence and write access. It will be created for you if missing. If it cannot be created or does not have proper write access, application deployment will fail.

dataverse.auth.password-reset-timeout-in-minutes

Users have 60 minutes to change their passwords by default. You can adjust this value here.

dataverse.db.name

The PostgreSQL database name to use for the Dataverse installation.

Defaults to `dataverse` (but the installer sets it to `dvndb`).

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_DB_NAME`.

See also *Database Persistence*.

dataverse.db.user

The PostgreSQL user name to connect with.

Defaults to `dataverse` (but the installer sets it to `dvnapp`).

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_DB_USER`.

dataverse.db.password

The PostgreSQL users password to connect with.

See *Secure Password Storage* to learn about options to securely store this password.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_DB_PASSWORD` (although you shouldn't use environment variables for passwords).

dataverse.db.host

The PostgreSQL server to connect to.

Defaults to `localhost`.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_DB_HOST`.

dataverse.db.port

The PostgreSQL server port to connect to.

Defaults to `5432`, the default PostgreSQL port.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_DB_PORT`.

dataverse.db.parameters

The PostgreSQL server connection parameters.

Defaults to *empty string*

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_DB_PARAMETERS`.

dataverse.solr.host

The hostname of a Solr server to connect to. Remember to restart / redeploy Dataverse after changing the setting (as with *:SolrHostColonPort*).

Defaults to `localhost`.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_SOLR_HOST`. Defaults to `solr`, when used with `mp.config.profile=ct` (*see below*).

dataverse.solr.port

The Solr server port to connect to. Remember to restart / redeploy Dataverse after changing the setting (as with *:SolrHostColonPort*).

Defaults to `8983`, the default Solr port.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_SOLR_PORT`.

dataverse.solr.core

The name of the Solr core to use for this Dataverse installation. Might be used to switch to a different core quickly. Remember to restart / redeploy Dataverse after changing the setting (as with *:SolrHostColonPort*).

Defaults to `collection1`.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_SOLR_CORE`.

dataverse.solr.protocol

The Solr server URL protocol for the connection. Remember to restart / redeploy Dataverse after changing the setting (as with *:SolrHostColonPort*).

Defaults to `http`, but might be set to `https` for extra secure Solr installations.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_SOLR_PROTOCOL`.

dataverse.solr.path

The path part of the Solr endpoint URL (e.g. `/solr/collection1` of `http://localhost:8389/solr/collection1`). Might be used to target a Solr API at non-default places. Remember to restart / redeploy Dataverse after changing the setting (as with *:SolrHostColonPort*).

Defaults to `/solr/${dataverse.solr.core}`, interpolating the core name when used. Make sure to include the variable when using it to configure your core name!

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_SOLR_PATH`.

dataverse.solr.min-files-to-use-proxy

Specifies when to use a smaller datafile proxy object for the purposes of dataset indexing. This can lower memory requirements and improve performance when reindexing large datasets (e.g. those with hundreds or thousands of files). (Creating the proxy may slightly slow indexing datasets with only a few files.)

This setting represents a number of files for which the datafile procy should be used. By default, this is set to Interger.MAX which disables using the proxy. A recommended value would be ~1000 but the optimal value may vary depending on details of your installation.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_SOLR_MIN_FILES_TO_USE_PROXY`.

dataverse.solr.concurrency.max-async-indexes

Maximum number of simultaneously running asynchronous dataset index operations.

Defaults to `4`.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_SOLR_CONCURRENCY_MAX_ASYNC_INDEXES`.

dataverse.rserve.host

Host name for Rserve, used for tasks that require use of R (to ingest RData files and to save tabular data as RData frames).

Defaults to `localhost`.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_RSERVE_HOST`.

dataverse.rserve.port

Port number for Rserve, used for tasks that require use of R (to ingest RData files and to save tabular data as RData frames).

Defaults to `6311` when not configured or no valid integer.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_RSERVE_PORT`.

dataverse.rserve.user

Username for Rserve, used for tasks that require use of R (to ingest RData files and to save tabular data as RData frames).

Defaults to `rserve`.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_RSERVE_USER`.

dataverse.rserve.password

Password for Rserve, used for tasks that require use of R (to ingest RData files and to save tabular data as RData frames).

Defaults to `rserve`.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_RSERVE_PASSWORD`.

dataverse.rserve.tempdir

Temporary directory used by Rserve (defaults to `/tmp/Rserv`). Note that this location is local to the host on which Rserv is running (specified in `dataverse.rserve.host` above). When talking to Rserve, Dataverse needs to know this location in order to generate absolute path names of the files on the other end.

Defaults to `/tmp/Rserv`.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_RSERVE_TEMPDIR`.

dataverse.dropbox.key

Dropbox provides a Chooser app, which is a Javascript component that allows you to upload files to your Dataverse installation from Dropbox. It is an optional configuration setting, which requires you to pass it an app key and configure the `:UploadMethods` database setting. For more information on setting up your Chooser app, visit https://www.dropbox.com/developers/chooser.

```
./asadmin create-jvm-options "-Ddataverse.dropbox.key={{YOUR_APP_KEY}}"
```

dataverse.path.imagemagick.convert

For overriding the default path to the `convert` binary from ImageMagick (`/usr/bin/convert`).

dataverse.dataAccess.thumbnail.image.limit

For limiting the size (in bytes) of thumbnail images generated from files. The default is 3000000 bytes (3 MB).

dataverse.dataAccess.thumbnail.pdf.limit

For limiting the size (in bytes) of thumbnail images generated from files. The default is 1000000 bytes (1 MB).

Legacy Single PID Provider: dataverse.pid.datacite.mds-api-url

Configure the base URL of the DataCite MDS API, used to mint and manage DOIs. Valid values are "https://mds.datacite.org" and "https://mds.test.datacite.org" (see also note below).

Out of the box, the installer configures your installation to use a DataCite REST Test API base URL (see DataCite's testing guide). You can delete it like this:

```
./asadmin delete-jvm-options '-Ddataverse.pid.datacite.mds-api-url=https\://mds.test.
datacite.org'
```

Then, to switch to the production DataCite base URL (see the DataCite MDS API Guide), you can issue the following command:

```
./asadmin create-jvm-options '-Ddataverse.pid.datacite.mds-api-url=https\://mds.datacite.
org'
```

Without setting an option, always defaults to testing API endpoint.

Notes:

- See also these related database settings below: *Legacy Single PID Provider: :DoiProvider*, *Legacy Single PID Provider: :Protocol*, *Legacy Single PID Provider: :Authority*, *Legacy Single PID Provider: :Shoulder*.
- Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_PID_DATACITE_MDS_API_URL`.
- This setting was formerly known as `doi.baseurlstring` and has been renamed. You should delete and re-add it.
- While using DataCite directly is recommended because it is tested by the Dataverse Project Team plus field tested with most installations, it is also possible to use a DataCite Client API as a proxy to DataCite. Since the launch of DataCite Fabrica in 2019, the only example by Australian National Data Services (ANDS) has been decommissioned.

Legacy Single PID Provider: dataverse.pid.datacite.rest-api-url

Configure the base URL endpoint of the DataCite REST API, used for *PIDs API* information retrieval and *Make Data Count*.

Valid values are "https://api.datacite.org" and "https://api.test.datacite.org". When unset, the default is the testing API endpoint.

Out of the box, the installer configures your installation to use a DataCite REST test base URL (see DataCite's testing guide). You can delete it like this:

```
./asadmin delete-jvm-options '-Ddataverse.pid.datacite.rest-api-url=https\://api.test.
datacite.org'
```

Then, to switch to the production DataCite base URL (see the DataCite REST API Guide), you can issue the following command:

```
./asadmin create-jvm-options '-Ddataverse.pid.datacite.rest-api-url=https\://api.
datacite.org'
```

Notes:

- See also these related database settings below: *:MDCLogPath*, *:DisplayMDCMetrics*.
- Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_PID_DATACITE_REST_API_URL`.
- This setting was formerly known as `doi.dataciterestapiurlstring` or `doi.mdcbaseurlstring` and has been renamed. You should delete these and re-add it (once) under the new name.

Legacy Single PID Provider: dataverse.pid.datacite.username

DataCite uses HTTP Basic authentication for Fabrica and their APIs. You need to provide the same credentials to Dataverse software to mint and manage DOIs for you.

Once you have a username from DataCite, you can enter it like this:

```
./asadmin create-jvm-options '-Ddataverse.pid.datacite.username=YOUR_USERNAME_HERE'
```

Notes:

- Used in conjuction with *Legacy Single PID Provider: dataverse.pid.datacite.mds-api-url*, *Legacy Single PID Provider: dataverse.pid.datacite.rest-api-url* and *Legacy Single PID Provider: dataverse.pid.datacite.password*.
- Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_PID_DATACITE_USERNAME`.
- This setting was formerly known as `doi.username` and has been renamed. You should delete and re-add it.

Legacy Single PID Provider: dataverse.pid.datacite.password

Once you have a password from your provider, you should create a password alias called *dataverse.pid.datacite.password* or use another method described at *Secure Password Storage* to safeguard it.

Notes:

- Used in conjuction with *Legacy Single PID Provider: dataverse.pid.datacite.mds-api-url*, *Legacy Single PID Provider: dataverse.pid.datacite.rest-api-url* and *Legacy Single PID Provider: dataverse.pid.datacite.username*.
- Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_PID_DATACITE_PASSWORD` (although you shouldn't use environment variables for passwords).

- This setting was formerly known as `doi.password` and has been renamed. You should delete the old JVM option and the wrapped password alias, then recreate as described above.

Legacy Single PID Provider: dataverse.pid.handlenet.key.path

Related to *Handle.Net PID provider usage*.

Provide an absolute path to a private key file authenticating requests to your Handle.Net server.

Handle.Net servers use a public key authentication method where the public key is stored in a handle itself and the matching private key is provided from this file. Typically, the absolute path ends like `handle/svr_1/admpriv.bin`. See also chapter 1.4 "Authentication" of the Handle.Net Technical Documentation

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_PID_HANDLENET_KEY_PATH`. This setting was formerly known as `dataverse.handlenet.admcredfile` and has been renamed. You should delete and re-add it.

Legacy Single PID Provider: dataverse.pid.handlenet.key.passphrase

Related to *Handle.Net PID provider usage*.

Provide a passphrase to decrypt the *private key file*. See *Secure Password Storage* for ways to do this securely.

The key file may (and should) be encrypted with a passphrase (used for encryption with AES-128). See also chapter 1.4 "Authentication" of the Handle.Net Technical Documentation

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_PID_HANDLENET_KEY_PASSPHRASE` (although you shouldn't use environment variables for passwords).

This setting was formerly known as `dataverse.handlenet.admprivphrase` and has been renamed. You should delete the old JVM option and the wrapped password alias, then recreate as shown for *Legacy Single PID Provider: dataverse.pid.datacite.password* but with this option as alias name.

Legacy Single PID Provider: dataverse.pid.handlenet.index

Related to *Handle.Net PID provider usage*.

Configure your *Handle.Net Index* to be used registering new persistent identifiers. Defaults to `300`.

Indices are used to separate concerns within the Handle system. To add data to an index, authentication is mandatory. See also chapter 1.4 "Authentication" of the Handle.Net Technical Documentation

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_PID_HANDLENET_INDEX`. This setting was formerly known as `dataverse.handlenet.index` and has been renamed. You should delete and re-add it.

Legacy Single PID Provider: dataverse.pid.permalink.base-url

When using *PermaLinks*, this setting can be used to configure an external resolver. Dataverse will associate a PermaLink PID with the URL: `<dataverse.pid.permalink.base-url>/citation?persistentId=perma:<permalink>`. The default value is your Dataverse site URL, which will result in PermaLinks correctly resolving to the appropriate dataset page.

To set this option, issue a command such as:

```
./asadmin create-jvm-options '-Ddataverse.pid.permalink.base-url=https\://localresolver.
yourdataverse.org'
```

See also these related database settings:

- *Legacy Single PID Provider: :Protocol*
- *Legacy Single PID Provider: :Authority*
- *Legacy Single PID Provider: :Shoulder*

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_PID_PERMALINK_BASE_URL`. This setting was formerly known as `perma.baseurlstring` and has been renamed. You should delete and re-add it.

Legacy Single PID Provider: dataverse.pid.ezid.api-url

The EZID DOI provider is likely not an option if you are not associated with California Digital Library (CDL) or Purdue University.

Defaults to `https://ezid.cdlib.org`.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_PID_EZID_API_URL`. This setting was formerly known as `doi.baseurlstring` and has been renamed. You should delete and re-add it.

Legacy Single PID Provider: dataverse.pid.ezid.username

The EZID DOI provider is likely not an option if you are not associated with California Digital Library (CDL) or Purdue University.

Works the same way as *Legacy Single PID Provider: dataverse.pid.datacite.username*, but for the EZID DOI provider.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_PID_EZID_USERNAME`.

This setting was formerly known as `doi.username` and has been renamed. You should delete and re-add it.

Legacy Single PID Provider: dataverse.pid.ezid.password

The EZID DOI provider is likely not an option if you are not associated with California Digital Library (CDL) or Purdue University.

Works the same way as *Legacy Single PID Provider: dataverse.pid.datacite.password*, but for the EZID DOI provider.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_PID_EZID_PASSWORD` (although you shouldn't use environment variables for passwords).

This setting was formerly known as `doi.password` and has been renamed. You should delete the old JVM option and the wrapped password alias, then recreate as shown for *Legacy Single PID Provider: dataverse.pid.datacite.password* but with the EZID alias name.

dataverse.timerServer

This JVM option is only relevant if you plan to run multiple Payara servers for redundancy. Only one Payara server can act as the dedicated timer server and for details on promoting or demoting a Payara server to handle this responsibility, see *Dataverse Installation Application Timers*.

dataverse.lang.directory

This JVM option is used to configure the path where all the language specific property files are to be stored. If this option is set then the English property file must be present in the path along with any other language property file. You can download language property files from https://github.com/GlobalDataverseCommunityConsortium/dataverse-language-packs

```
./asadmin create-jvm-options '-Ddataverse.lang.directory=PATH_LOCATION_HERE'
```

If this value is not set, by default, a Dataverse installation will read the English language property files from the Java Application.

See also *Internationalization*.

dataverse.files.hide-schema-dot-org-download-urls

Please note that this setting is experimental.

By default, download URLs to files will be included in Schema.org JSON-LD output. To prevent these URLs from being included in the output, set `dataverse.files.hide-schema-dot-org-download-urls` to true as in the example below.

```
./asadmin create-jvm-options '-Ddataverse.files.hide-schema-dot-org-download-urls=true'
```

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_FILES_HIDE_SCHEMA_DOT_ORG_DOWNLOAD_URLS`.

For more on Schema.org JSON-LD, see the *Metadata Export* section of the Admin Guide.

dataverse.useripaddresssourceheader

Make sure to read the section about the *Security Implications* of using this option earlier in the guide!

If set, specifies an HTTP Header such as X-Forwarded-For to use to retrieve the user's IP address. For example:

```
./asadmin create-jvm-options '-Ddataverse.useripaddresssourceheader=X-Forwarded-For'
```

This setting is useful in cases such as running your Dataverse installation behind load balancers where the default option of getting the Remote Address from the servlet isn't correct (e.g. it would be the load balancer IP address). Note that unless your installation always sets the header you configure here, this could be used as a way to spoof the user's address. Allowed values are:

```
"X-Forwarded-For",
"Proxy-Client-IP",
"WL-Proxy-Client-IP",
"HTTP_X_FORWARDED_FOR",
"HTTP_X_FORWARDED",
"HTTP_X_CLUSTER_CLIENT_IP",
"HTTP_CLIENT_IP",
"HTTP_FORWARDED_FOR",
```

(continues on next page)

(continued from previous page)

```
"HTTP_FORWARDED",
"HTTP_VIA",
"REMOTE_ADDR"
```

dataverse.person-or-org.assume-comma-in-person-name

Please note that this setting is experimental.

The Schema.org metadata and OpenAIRE exports and the Schema.org metadata included in DatasetPages try to infer whether each entry in the various fields (e.g. Author, Contributor) is a Person or Organization. If you are sure that users are following the guidance to add people in the recommended family name, given name order, with a comma, you can set this true to always assume entries without a comma are for Organizations. The default is false.

```
./asadmin create-jvm-options '-Ddataverse.person-or-org.assume-comma-in-person-name=true'
```

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_PERSON_OR_ORG_ASSUME_COMMA_IN_PERSON_NAME`.

Note: This setting was previously called *dataverse.personOrOrg.assumeCommaInPersonName*, which is still available as an alias for backwards compatiblity.

dataverse.person-or-org.org-phrase-array

Please note that this setting is experimental.

The Schema.org metadata and OpenAIRE exports and the Schema.org metadata included in DatasetPages try to infer whether each entry in the various fields (e.g. Author, Contributor) is a Person or Organization. If you have examples where an organization name is being inferred to belong to a person, you can use this setting to force it to be recognized as an organization. The value is expected to be a comma-separated list of strings. Any name that contains one of the strings is assumed to be an organization. For example, "Project" is a word that is not otherwise associated with being an organization.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_PERSON_OR_ORG_ORG_PHRASE_ARRAY`.

Note: This setting was previously called *dataverse.personOrOrg.orgPhraseArray* and expected a JsonArray of strings. Please update both the name and value format if using the old setting.

dataverse.api.signing-secret

Context: Dataverse has the ability to create "Signed URLs" for it's API calls. Using a signed URLs is more secure than providing API tokens, which are long-lived and give the holder all of the permissions of the user. In contrast, signed URLs are time limited and only allow the action of the API call in the URL. See *Authorization Options* and *Request Signed URL* for more details.

The key used to sign a URL is created from the API token of the creating user plus a server-side signing secret. By default, the secret is generated automatically at startup and kept only in memory: no configuration is needed, but signed URLs do not survive a server restart, and on a multi-server installation each server signs with its own secret. Set this option (to a value of at least 36 characters - shorter ones are ignored with a warning) to use a persistent secret instead: previously issued signed URLs then stay valid across restarts, and all servers sign and validate with the same key.

WARNING: *Since the signing-secret is sensitive, you should treat it like a password. See Secure Password Storage to learn about ways to safeguard it.*

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_API_SIGNING_SECRET` (although you shouldn't use environment variables for passwords) .

dataverse.api.allow-incomplete-metadata

When enabled, dataset with incomplete metadata can be submitted via API for later corrections. See *Create a Dataset in a Dataverse Collection* for details.

Defaults to `false`.

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_API_ALLOW_INCOMPLETE_METADATA`. Will accept `[tT][rR][uU][eE]|1|[oO][nN]` as "true" expressions.

dataverse.api.blocked.endpoints

A comma-separated list of API endpoints that should be blocked. A minimal example that blocks endpoints for security reasons:

```
./asadmin create-jvm-options '-Ddataverse.api.blocked.endpoints=admin,builtin-users'
```

Another example:

```
./asadmin create-jvm-options '-Ddataverse.api.blocked.endpoints=admin,builtin-users,
datasets/:persistentId/versions/:versionId/files,files/:id'
```

Defaults to an empty string (no endpoints blocked), but, in almost all cases, should include at least `admin, builtin-users` as a security measure.

For more information on API blocking, see *Blocking API Endpoints* in the Admin Guide.

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_API_BLOCKED_ENDPOINTS`.

dataverse.api.blocked.policy

Specifies how to treat blocked API endpoints. Valid values are:

- `drop`: Blocked requests are dropped (default).
- `localhost-only`: Blocked requests are only allowed from localhost.
- `unblock-key`: Blocked requests are allowed if they include a valid unblock key.

For example:

```
./asadmin create-jvm-options '-Ddataverse.api.blocked.policy=localhost-only'
```

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_API_BLOCKED_POLICY`.

> **Note**
>
> This setting will be ignored unless the *dataverse.api.blocked.endpoints* and, for the unblock-key policy, the *dataverse.api.blocked.key* are also set. Otherwise the deprecated *:BlockedApiPolicy (Deprecated)* will be used

dataverse.api.blocked.key

When the blocked API policy is set to `unblock-key`, this setting specifies the key that allows access to blocked endpoints. For example:

```
./asadmin create-jvm-options '-Ddataverse.api.blocked.key=your-secret-key-here'
```

WARNING: *Since the blocked API key is sensitive, you should treat it like a password. See Secure Password Storage to learn about ways to safeguard it.*

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_API_BLOCKED_KEY` (although you shouldn't use environment variables for sensitive information).

> **Note**
>
> This setting will be ignored unless the *dataverse.api.blocked.policy* is set to `unblock-key`. Otherwise the deprecated *:BlockedApiKey (Deprecated)* will be used

dataverse.legacy.api-response-message-style

Opt-out of no longer nesting an object in the "message" field, carrying the actual notification in its "message" field. Enabling this will re-activate the legacy message style using `{"message":{"message":"..."}}`, instead of the aligned format `{"message": "..."}`.

This option is provided as a temporary workaround for integrations that may have implemented workarounds for the buggy behavior. The following endpoints are affected:

- `POST /api/datasets/{id}/add` (just the duplicate file warning)
- `PUT /api/admin/settings`
- `PUT /api/dataverses/{id}`
- `PUT /api/dataverses/{id}/inputLevels`
- `POST /api/admin/savedsearches`
- `PUT /api/harvest/clients/{nickName}`
- `PUT /api/harvest/server/oaisets/{specname}`

Please update your integrations to expect the corrected message format and deactivate this setting. In a future version of Dataverse, the legacy format is expected to be removed completely. See also *dataverse.feature.unify-api-response-message-style*.

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_LEGACY_API_RESPONSE_MESSAGE_STYLE`.

dataverse.ui.show-validity-label-when-published

Even when you do not allow incomplete metadata to be saved in dataverse, some metadata may end up being incomplete, e.g., after making a metadata field mandatory. Datasets where that field is not filled out, become incomplete, and therefore can be labeled with the `incomplete metadata` label. By default, this label is only shown for draft datasets and published datasets that the user can edit. This option can be disabled by setting it to `false` where only draft datasets with incomplete metadata will have that label. When disabled, all published dataset will not have that label. Note that you need to reindex the datasets after changing the metadata definitions. Reindexing will update the labels and other dataset information according to the new situation.

When enabled (by default), published datasets with incomplete metadata will have an `incomplete metadata` label attached to them, but only for the datasets that the user can edit. You can list these datasets, for example, with the validity of metadata filter shown in "My Data" page that can be turned on by enabling the *dataverse.ui.show-validity-filter* option.

Defaults to `true`.

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_API_SHOW_LABEL_FOR_INCOMPLETE_WHEN_PUBLISHED`. Will accept `[tT][rR][uU][eE]|1|[oO][nN]` as "true" expressions.

dataverse.ui.show-curation-status-to-all

By default the curation status assigned to a draft dataset versioncan only be seen by those who can publish it. When this flag is true, anyone who can see the draft dataset can see the assigned status. These users will also get notifications/emails about changes to the status. See *:AllowedCurationLabels* and the *Managing Datasets and Dataverse Collections* section for more information about curation status.

Defaults to `false`.

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_API_SHOW_CURATION_STATUS_TO_ALL`. Will accept `[tT][rR][uU][eE]|1|[oO][nN]` as "true" expressions.

dataverse.signposting.level1-author-limit

See *Signposting* for details.

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_SIGNPOSTING_LEVEL1_AUTHOR_LIMIT`.

dataverse.signposting.level1-item-limit

See *Signposting* for details.

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_SIGNPOSTING_LEVEL1_ITEM_LIMIT`.

dataverse.mail.system-email

This is the email address that "system" emails are sent from such as password reset links, notifications, etc. It replaces the database setting *:SystemEmail* since Dataverse 6.2.

WARNING: Your Dataverse installation will not send mail without this setting in place.

Note that only the email address is required, which you can supply without the < and > signs, but if you include the text, it's the way to customize the name of your support team, which appears in the "from" address in emails as well as in help text in the UI. If you don't include the text, the installation name (see *Branding Your Installation*) will appear in the "from" address. In case you want your system email address to of no-reply style, have a look at *dataverse.mail.support-email* setting, too.

Please note that if you're having any trouble sending email, you can refer to "Troubleshooting" under *Installation*.

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_MAIL_SYSTEM_EMAIL`.

See also *SMTP/Email Configuration*.

dataverse.mail.support-email

This provides an email address distinct from the *dataverse.mail.system-email* that will be used as the email address for Contact Forms and Feedback API. This address is used as the To address when the Contact form is launched from the Support entry in the top navigation bar and, if configured via *dataverse.mail.cc-support-on-contact-email*, as a CC address when the form is launched from a Dataverse/Dataset Contact button. This allows configuration of a no-reply email address for *dataverse.mail.system-email* while allowing feedback to go to/be cc'd to the support email address, which would normally accept replies. If not set, the *dataverse.mail.system-email* is used for the feedback API/contact form email.

Note that only the email address is required, which you can supply without the < and > signs, but if you include the text, it's the way to customize the name of your support team, which appears in the "from" address in emails as well as in help text in the UI. If you don't include the text, the installation name (see *Branding Your Installation*) will appear in the "from" address.

Also note that the support email address is used at the end of notification mails where it states; "contact us for support at", followed by the support mail address if configured and the system email otherwise.

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_MAIL_SUPPORT_EMAIL`.

See also *SMTP/Email Configuration*.

dataverse.mail.cc-support-on-contact-email

If this boolean setting is true, the contact forms and feedback API will cc the system (`dataverse.mail.support-mail` if set, `dataverse.mail.system-email` if not) when sending email to the collection, dataset, or datafile contacts. A CC line is added to the contact form when this setting is true so that users are aware that the cc will occur. The default is false.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_MAIL_CC_SUPPORT_ON_CONTACT_EMAIL`.

See also *SMTP/Email Configuration*.

dataverse.mail.debug

When this boolean setting is true, sending an email will generate more verbose logging, enabling you to analyze mail delivery malfunctions. Defaults to `false`.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_MAIL_DEBUG`.

See also *SMTP/Email Configuration*.

dataverse.mail.mta.*

The following options allow you to configure a target Mail Transfer Agent (MTA) to be used for sending emails to users. Be advised: as the mail server connection (session) is cached once created, you need to restart Payara when applying configuration changes.

All can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_MAIL_MTA_HOST`. (For environment variables: simply replace "." and "-" with "_" and write as all caps.)

The following table describes the most important settings commonly used.

Setting Key	Description	Default Value
`dataverse.mail.mta.host`	The SMTP server to connect to.	*No default*
`dataverse.mail.mta.port`	The SMTP server port to connect to. (Common are `25` for plain, `587` for SSL, `465` for legacy SSL)	`25`
`dataverse.mail.mta.ssl.enable`	Enable if your mail provider uses SSL.	`false`
`dataverse.mail.mta.auth`	If `true`, attempt to authenticate the user using the AUTH command.	`false`
`dataverse.mail.mta.user`	The username to use in an AUTH command.	*No default*
`dataverse.mail.mta.password`	The password to use in an AUTH command. (Might be a token when using XOAUTH2 mechanism)	*No default*
`dataverse.mail.mta.allow-utf8-a`	If set to `true`, UTF-8 strings are allowed in message headers, e.g., in addresses. This should only be set if the mail server also supports UTF-8. (Quoted from Jakarta Mail Javadoc) Setting to `false` will also make mail address validation in UI/API fail on UTF-8 chars.	`true`

WARNING: *For security of your password use only safe ways to store and access it. See Secure Password Storage to learn about your options.*

Find below a list of even more options you can use to configure sending mails. Detailed description for every setting can be found in the table included within the Jakarta Mail Documentation. (Simply replace `dataverse.mail.mta.` with `mail.smtp.`.)

- **Timeouts:**
  `dataverse.mail.mta.connectiontimeout`, `dataverse.mail.mta.timeout`, `dataverse.mail.mta.writetimeout`
- **SSL/TLS:**
  `dataverse.mail.mta.starttls.enable`, `dataverse.mail.mta.starttls.required`, `dataverse.mail.mta.ssl.checkserveridentity`, `dataverse.mail.mta.ssl.trust`, `dataverse.mail.mta.ssl.protocols`, `dataverse.mail.mta.ssl.ciphersuites`
- **Proxy Connection:**
  `dataverse.mail.mta.proxy.host`, `dataverse.mail.mta.proxy.port`, `dataverse.mail.mta.proxy.user`, `dataverse.mail.mta.proxy.password`, `dataverse.mail.mta.socks.host`, `dataverse.mail.mta.socks.port`
- **SMTP EHLO command details:**
  `dataverse.mail.mta.ehlo`, `dataverse.mail.mta.localhost`, `dataverse.mail.mta.localaddress`, `dataverse.mail.mta.localport`
- **Authentication details:**
  `dataverse.mail.mta.auth.mechanisms`, `dataverse.mail.mta.auth.login.disable`, `dataverse.mail.mta.auth.plain.disable`, `dataverse.mail.mta.auth.digest-md5.disable`, `dataverse.mail.mta.auth.ntlm.disable`, `dataverse.mail.mta.auth.xoauth2.disable`, `dataverse.mail.mta.auth.ntlm.domain`, `dataverse.mail.mta.auth.ntlm.flag`, `dataverse.mail.mta.sasl.enable`, `dataverse.mail.mta.sasl.usecanonicalhostname`,

dataverse.mail.mta.sasl.mechanisms, dataverse.mail.mta.sasl.authorizationid, dataverse.mail.mta.sasl.realm

- **Miscellaneous:**
  dataverse.mail.mta.allow8bitmime, dataverse.mail.mta.submitter, dataverse.mail.mta.dsn.notify, dataverse.mail.mta.dsn.ret, dataverse.mail.mta.sendpartial, dataverse.mail.mta.quitwait, dataverse.mail.mta.quitonsessionreject, dataverse.mail.mta.userset, dataverse.mail.mta.noop.strict, dataverse.mail.mta.mailextension

See also *SMTP/Email Configuration*.

dataverse.ui.allow-review-for-incomplete

Determines if dataset submitted via API with incomplete metadata (for later corrections) can be submitted for review from the UI.

Defaults to `false`.

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_UI_ALLOW_REVIEW_FOR_INCOMPLETE`. Will accept `[tT][rR][uU][eE]|1|[oO][nN]` as "true" expressions.

dataverse.ui.show-validity-filter

When enabled, the filter for validity of metadata is shown in "My Data" page. **Note:** When you wish to use this filter, you must reindex the datasets first, otherwise datasets with valid metadata will not be shown in the results.

Defaults to `false`.

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_UI_SHOW_VALIDITY_FILTER`. Will accept `[tT][rR][uU][eE]|1|[oO][nN]` as "true" expressions.

dataverse.spi.exporters.directory

For some background, see *External Metadata Exporters* and *Inventory of External Exporters*.

This JVM option is used to configure the file system path where external exporter JARs should be loaded from. For example:

```
./asadmin create-jvm-options '-Ddataverse.spi.exporters.directory=/var/lib/dataverse/
exporters'
```

If this value is set, Dataverse will examine all JARs in the specified directory and will use them to add new metadata export formats or (if the machine-readable name used in *Export Metadata of a Dataset in Various Formats* is the same) replace built-in metatadata export formats.

If this value is not set (the default), Dataverse will not load any external exporters.

If you place a new JAR in this directory, you must restart Payara for Dataverse to load it.

If the JAR is for an exporter that replaces built-in format, you must delete the cached exports and/or use a reExport API call (see *Batch Exports Through the API*) for the new format to be visible for existing datasets.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_SPI_EXPORTERS_DIRECTORY`.

dataverse.netcdf.geo-extract-s3-direct-upload

This setting was added to keep S3 direct upload lightweight. When that feature is enabled and you still want NetCDF and HDF5 files to go through metadata extraction of a Geospatial Bounding Box (see *NetCDF and HDF5*), which requires the file to be downloaded from S3 in this scenario, make this setting true.

See also *Features that are Disabled if S3 Direct Upload is Enabled*.

dataverse.storageuse.disable-storageuse-increments

This setting serves the role of an emergency "kill switch" that will disable maintaining the real time record of storage use for all the datasets and collections in the database. Because of the experimental nature of this feature (see *Storage Quotas for Collections*) that hasn't been used in production setting as of this release, v6.1 this setting is provided in case these updates start causing database race conditions and conflicts on a busy server.

dataverse.auth.oidc.*

Provision a single *OpenID Connect authentication provider* using MicroProfile Config. You can find a list of all available options at *Provision via JVM Options*.

dataverse.files.guestbook-at-request

This setting enables functionality to allow guestbooks to be displayed when a user requests access to a restricted data file(s) or when a file is downloaded (the historic default). Providing a true/false value for this setting enables the functionality and provides a global default. The behavior can also be changed at the collection level via the user interface and by a superuser for a give dataset using the API.

See also *Configure When a Dataset Guestbook Appears (If Enabled)* in the API Guide, and .

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_FILES_GUESTBOOK_AT_REQUEST`.

dataverse.bagit.sourceorg.name

The name for your institution that you'd like to appear in bag-info.txt. See *Configuring bag-info.txt*.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_BAGIT_SOURCEORG_NAME`.

dataverse.bagit.sourceorg.address

The mailing address for your institution that you'd like to appear in bag-info.txt. See *Configuring bag-info.txt*. The example in https://datatracker.ietf.org/doc/html/rfc8493 uses commas as separators: `1 Main St., Cupertino, California, 11111`.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_BAGIT_SOURCEORG_ADDRESS`.

dataverse.bagit.sourceorg.email

The email for your institution that you'd like to appear in bag-info.txt. See *Configuring bag-info.txt*.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_BAGIT_SOURCEORG_EMAIL`.

dataverse.bagit.archive-on-version-update

Indicates whether archival bag creation should be triggered (if configured) when a version is updated and was already successfully archived, i.e via the Update-Current-Version publication option. Setting the flag true only works if the archiver being used supports deleting existing archival bags.

dataverse.files.globus-monitoring-server

This setting is required in conjunction with the *dataverse.feature.globus-use-experimental-async-framework* feature flag. Setting it to true designates the Dataverse instance to serve as the dedicated polling server. It is needed so that the new framework can be used in a multi-node installation.

dataverse.csl.common-styles

This setting allows admins to highlight a few of the 1000+ CSL citation styles available from the dataset page. The value should be a comma-separated list of styles. These will be listed above the alphabetical list of all styles in the "View Styled Citations" pop-up. The default value when not set is "chicago-author-date, ieee".

dataverse.localcontexts.url

> **Note**
>
> For more information about LocalContexts integration, see *LocalContexts Integration*.

The URL for the Local Contexts Hub API.

Example: `https://localcontextshub.org/`
The sandbox URL `https://sandbox.localcontextshub.org/` can be used for testing.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_LOCALCONTEXTS_URL`.

dataverse.localcontexts.api-key

The API key for accessing the Local Contexts Hub.

Example: `your_api_key_here`
It's recommended to use a password alias for this setting, as described in the *Secure Password Storage* section.

Can also be set via *MicroProfile Config API* sources, e.g. the environment variable `DATAVERSE_LOCALCONTEXTS_API_KEY`.

dataverse.search.services.directory

Experimental. See *Search Services*.

dataverse.search.default-service

Experimental. See *Search Services*.

dataverse.cors.origin

Allowed origins for CORS requests. See also *Cross-Origin Resource Sharing (CORS)*.

Default: *not configured*

Warning

If this setting is not explicitly configured, no CORS headers at all are added to responses.
The default policy (see all CORS related settings) is still being enforced!

Type	Value/Ex	Description
Wild-card	*	• Allow access from all origins. • Response header echoes `Access-Control-Allow-Origin: *` • Browsers will not allow credentialed requests with this setting.
List of Origins	`https://example.org`, `https://example.com`	• Comma separated, white space ignored. • Single matching request `Origin` header echoed as response header `Access-Control-Allow-Origin`. • `Vary: Origin` header added to support correct proxy/CDN caching. • Use `${dataverse.siteurl}` to dynamically add the installation's URL to the list.

Example:

```
./asadmin create-jvm-options '-Ddataverse.cors.origin=https://example.com,https://
subdomain.example.com'
```

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_CORS_ORIGIN`.

dataverse.cors.methods

Allowed HTTP methods for CORS requests as a comma separated list. Whitespace is ignored.

Default: `GET,POST,OPTIONS,PUT,DELETE`

Example:

```
./asadmin create-jvm-options '-Ddataverse.cors.methods=GET,POST,OPTIONS'
```

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_CORS_METHODS`.

dataverse.cors.headers.allow

Allowed headers for CORS requests as a comma separated list. Whitespace is ignored.

Default: `Accept, Content-Type, X-Dataverse-key, Range`

Example:

```
./asadmin create-jvm-options '-Ddataverse.cors.headers.allow=Accept,Content-Type,
X-Custom-Header'
```

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_CORS_HEADERS_ALLOW`.

dataverse.cors.headers.expose

Headers to expose in CORS responses as a comma separated list. Whitespace is ignored.

Default: `Accept-Ranges, Content-Range, Content-Encoding`

Example:

```
./asadmin create-jvm-options '-Ddataverse.cors.headers.expose=Accept-Ranges,
Content-Range,X-Custom-Header'
```

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_CORS_HEADERS_EXPOSE`.

dataverse.api.mdc.min-delay-ms

Minimum delay in milliseconds between Make Data Count (MDC) API requests from the /api/admin/makeDataCount/{id}/updateCitationsForDataset api. This setting helps prevent overloading the MDC service by enforcing a minimum time interval between consecutive requests. If a request arrives before this interval has elapsed since the previous request, it will be rate-limited.

Default: `0` (no delay enforced)

Example: `dataverse.api.mdc.min-delay-ms=100` (enforces a minimum 100ms delay between MDC API requests)

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_API_MDC_MIN_DELAY_MS`.

dataverse.legacy.schemaorg-in-html-head

Instead of Croissant, use the legacy format (Schema.org JSON-LD) in the head of dataset landing pages by setting `dataverse.legacy.schemaorg-in-html-head=true`. See *Croissant Metadata in the <head> of Dataset Landing Pages*.

Can also be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_LEGACY_SCHEMAORG_IN_HTML_HEAD`.

Linked Data Notifications (LDN) Allowed Hosts

Dataverse supports receiving LDN notifications via the /api/inbox endpoint. The dataverse.ldn.allowed-hosts allows you to specify the list of host IP addresses from which LDN notifications can be received, or * to receive messages from anywhere. Note that since the Inbox endpoint does not require authentication, allowing un-trusted hosts via * is not recommended for production.

Example: `dataverse.ldn.allowed-hosts=*` Example: `dataverse.ldn.allowed-hosts=172.16.234.56, 172.16.234.57`

COAR Notify Relationship Announcement Notify Superusers Only

When Dataverse receives an LDN message conforming to the COAR Notify Relationship Announcement format and the message is about a dataset hosted in the installation, Dataverse will send an notification to users who have permission to publish the dataset. This can instead be restricted to only superusers who can publish the dataset using this option.

Example: `dataverse.coar-notify.relationship-announcement.notify-superusers-only=true`

dataverse.bagit.zip.holey
: A boolean that, if true, will cause the BagIt archiver to create a "holey" bag. In a holey bag, files that are not included in the bag are listed in the `fetch.txt` file with a URL from which they can be downloaded. This is used in conjunction with `dataverse.bagit.zip.max-file-size` and/or `dataverse.bagit.zip.max-data-size`. Default: false.

dataverse.bagit.zip.max-data-size
: The maximum total (uncompressed) size of data files (in bytes) to include in a BagIt zip archive. If the total size of the dataset files exceeds this limit, files will be excluded from the zipped bag (starting from the largest) until the total size is under the limit. Excluded files will be handled as defined by `dataverse.bagit.zip.holey` - just listed if that setting is true or being transferred separately and placed next to the zipped bag. When not set, there is no limit.

dataverse.bagit.zip.max-file-size
: The maximum (uncompressed) size of a single file (in bytes) to include in a BagIt zip archive. Any file larger than this will be excluded. Excluded files will be handled as defined by `dataverse.bagit.zip.holey` - just listed if that setting is true or being transferred separately and placed next to the zipped bag. When not set, there is no limit.

6.5.21 Feature Flags

Certain features might be deactivated because they are experimental and/or opt-in capabilities. If you want to enable these, please find all known feature flags below. Any of these flags can be activated using a boolean value (case-insensitive, one of "true", "1", "YES", "Y", "ON") for the setting.

The default status, as long there is not any other information, is Off.

Note: Feature flags can be set via any supported MicroProfile Config API source, e.g. the environment variable `DATAVERSE_FEATURE_XXX` (e.g. `DATAVERSE_FEATURE_API_SESSION_AUTH=1`). These environment variables can be set in your shell before starting Payara. If you are using *Docker for development*, you can set them in the docker compose file.

To check the status of feature flags via API, see *List All Feature Flags* in the API Guide.

dataverse.feature.api-session-auth

Enables API authentication via session cookie (JSESSIONID). **Caution: Enabling this feature flag exposes the installation to CSRF risks!** We expect this feature flag to be temporary (only used by frontend developers, see #9063) and for the feature to be removed in the future.

dataverse.feature.api-bearer-auth

Enables API authentication via Bearer Token.

dataverse.feature.api-bearer-auth-provide-missing-claims

Enables sending missing user claims in the request JSON provided during OIDC user registration, when these claims are not returned by the identity provider and are required for registration. This feature only works when the feature flag `api-bearer-auth` is also enabled. **Caution: Enabling this feature flag exposes the installation to potential user impersonation issues.**

dataverse.feature.api-bearer-auth-handle-tos-acceptance-in-idp

Specifies that Terms of Service acceptance is handled by the IdP, eliminating the need to include ToS acceptance boolean parameter (termsAccepted) in the OIDC user registration request body. This feature only works when the feature flag `api-bearer-auth` is also enabled.

dataverse.feature.api-bearer-auth-use-builtin-user-on-id-match

Allows the use of a built-in user account when an identity match is found during API bearer authentication. This feature enables automatic association of an incoming IdP identity with an existing built-in user account, bypassing the need for additional user registration steps. This feature only works when the feature flag `api-bearer-auth` is also enabled. **Caution: Enabling this flag could result in impersonation risks if (and only if) used with a misconfigured IdP.**

dataverse.feature.api-bearer-auth-use-shib-user-on-id-match

Allows the use of a Shibboleth user account when an identity match is found during API bearer authentication. This feature enables automatic association of an incoming IdP identity with an existing Shibboleth user account, bypassing the need for additional user registration steps. This feature only works when the feature flag `api-bearer-auth` is also enabled. **Caution: Enabling this flag could result in impersonation risks if (and only if) used with a misconfigured IdP.**

dataverse.feature.api-bearer-auth-use-oauth-user-on-id-match

Allows the use of an OAuth user account (GitHub, Google, or ORCID) when an identity match is found during API bearer authentication. This feature enables automatic association of an incoming IdP identity with an existing OAuth user account, bypassing the need for additional user registration steps. This feature only works when the feature flag `api-bearer-auth` is also enabled. **Caution: Enabling this flag could result in impersonation risks if (and only if) used with a misconfigured IdP.**

dataverse.feature.unify-api-response-message-style

When activated, the "message" in API responses will no longer be nested in the "data" field. For any response carrying a notification, these will be found within a top-level "message" field of the JSON returned. This affects about 230 endpoints and is likely to break existing integrations and clients. It is mandatory to test instance clients and integrations thoroughly and it is not recommended to be used in production. In a future Dataverse version, the (currently) experimental response message style will be made the only supported one.

See also *dataverse.legacy.api-response-message-style*.

dataverse.feature.do-not-assume-default-license

When creating a dataset via API, if neither a license nor any terms of use are provided, the system normally assigns the default license. If this feature flag is enabled, no license is assigned (and no terms) in this case.

Defaults to `false`.

dataverse.feature.avoid-expensive-solr-join

Changes the way Solr queries are constructed for public content (published Collections, Datasets and Files). It removes a very expensive Solr join on all such documents, improving overall performance, especially for large instances under heavy load. Before this feature flag is enabled, the corresponding indexing feature (see next feature flag) must be turned on and a full reindex performed (otherwise public objects are not going to be shown in search results). See *Solr Search Index*.

dataverse.feature.add-publicobject-solr-field

Adds an extra boolean field *PublicObject_b:true* for public content (published Collections, Datasets and Files). Once reindexed with these fields, we can rely on it to remove a very expensive Solr join on all such documents in Solr queries, significantly improving overall performance (by enabling the feature flag above, *avoid-expensive-solr-join*). These two flags are separate so that an instance can reindex their holdings before enabling the optimization in searches, thus avoiding having their public objects temporarily disappear from search results while the reindexing is in progress.

dataverse.feature.reduce-solr-deletes

Avoids deleting and recreating solr documents for dataset files when reindexing.

dataverse.feature.disable-return-to-author-reason

Removes the reason field in the *Publish/Return To Author* dialog that was added as a required field in v6.2 and makes the reason an optional parameter in the *Return a Dataset to Author* API call.

dataverse.feature.disable-dataset-thumbnail-autoselect

Turns off automatic selection of a dataset thumbnail from image files in that dataset. When set to `On`, a user can still manually pick a thumbnail image or upload a dedicated thumbnail image.

dataverse.feature.globus-use-experimental-async-framework

Activates a new experimental implementation of Globus polling of ongoing remote data transfers that does not rely on the instance staying up continuously for the duration of the transfers and saves the state information about Globus upload requests in the database. Added in v6.4; extended in v6.6 to cover download transfers, in addition to uploads. Affects *:GlobusPollingInterval*. Note that the JVM option *dataverse.files.globus-monitoring-server* described above must also be enabled on one (and only one, in a multi-node installation) Dataverse instance.

dataverse.feature.index-harvested-metadata-source

Index the nickname or the source name (See the optional `sourceName` field in *Create a Harvesting Client*) of the harvesting client as the "metadata source" of harvested datasets and files. If enabled, the Metadata Source facet will show separate groupings of the content harvested from different sources (by harvesting client nickname or source name) instead of the default behavior where there is one "Harvested" grouping for all harvested content.

dataverse.feature.enable-version-note

Turns on the ability to add/view/edit/delete per-dataset-version notes intended to provide *Data Provenance* information about why the dataset/version was created.

dataverse.feature.shibboleth-use-wayfinder

This flag allows an instance to use Shibboleth with InCommon federation services. Our original Shibboleth implementation that relies on DiscoFeed can no longer be used since InCommon discontinued their old-style metadata feed. An alternative mechanism had to be implemented in order to use WayFinder service, their recommended replacements, instead.

dataverse.feature.shibboleth-use-localhost

A Shibboleth-using Dataverse instance needs to make network calls to the locally-running `shibd` service. The default behavior is to use the address configured via the `siteUrl` setting. There are however situations (firewalls, etc.) where localhost would be preferable.

dataverse.feature.add-local-contexts-permission-check

Adds a permission check to ensure that the user calling the /api/localcontexts/datasets/{id} API can edit the dataset with that id. This is currently the only use case - see https://github.com/gdcc/dataverse-external-vocab-support/tree/main/packages/local_contexts. The flag adds additional security to stop other uses, but would currently have to be used in conjunction with the api-session-auth feature flag (the security implications of which have not been fully investigated) to still allow adding Local Contexts metadata to a dataset.

dataverse.feature.enable-pid-failure-log

Turns on creation of a monthly log file (logs/PIDFailures_<yyyy-MM>.log) showing failed requests for dataset/file PIDs. Can be used directly or with scripts at https://github.com/gdcc/dataverse-recipes/python/pid_reports to alert admins.

dataverse.feature.role-assignment-history

Turns on tracking/display of role assignments and revocations for collections, datasets, and files

dataverse.feature.only-update-datacite-when-needed

Only contact DataCite to update a DOI after checking to see if DataCite has outdated information (for efficiency, lighter load on DataCite, especially when using file DOIs).

dataverse.feature.require-embargo-reason

Require an embargo reason when a user creates an embargo on one or more files. See *Embargoes*.

dataverse.feature.croissant-with-local-reviews

Have the croissant and croissantSlim metadata export formats include an extra "reviews" array if local reviews exist. See *Croissant Metadata in the <head> of Dataset Landing Pages*, *Review Datasets*, *Creating a Review Dataset*, and *List Reviews*.

dataverse.feature.allow-locally-fair-data

Allows support for Locally FAIR collections and datasets. When enabled, selected content can remain visible only to authorized users or groups within a Dataverse installation. See *Locally FAIR* for more information.

6.5.22 Application Server Settings

http.request-timeout-seconds

To facilitate large file upload and download, the Dataverse Software installer bumps the Payara **server-config.network-config.protocols.protocol.http-listener-1.http.request-timeout-seconds** setting from its default 900 seconds (15 minutes) to 1800 (30 minutes). Should you wish to shorten or lengthen this window, issue for example:

```
./asadmin set server-config.network-config.protocols.protocol.http-listener-1.http.
request-timeout-seconds=3600
```

and restart Payara to apply your change.

http.cookie-same-site-value

See *SameSite Cookie Attribute* for context.

The Dataverse installer configures the Payara **server-config.network-config.protocols.protocol.http-listener-1.http.cookie-same-site-value** setting to "Lax". From Payara's documentation, the other possible values are "Strict" or "None". To change this to "Strict", for example, you could run the following command…

```
./asadmin set server-config.network-config.protocols.protocol.http-listener-1.http.
cookie-same-site-value=Strict
```

… and restart Payara to apply your change.

http.cookie-same-site-enabled

See *SameSite Cookie Attribute* for context.

The Dataverse installer configures the Payara **server-config.network-config.protocols.protocol.http-listener-1.http.cookie-same-site-enabled** setting to true. To change this to false, you could run the following command…

```
./asadmin set server-config.network-config.protocols.protocol.http-listener-1.http.
cookie-same-site-enabled=true
```

… and restart Payara to apply your change.

mp.config.profile

MicroProfile Config 2.0 defines the concept of "profiles". They can be used to change configuration values by context. This is used in Dataverse to change some configuration defaults when used inside container context rather classic installations.

As per the spec, you will need to set the configuration value `mp.config.profile` to `ct` as early as possible. This is best done with a system property:

```
./asadmin create-system-properties 'mp.config.profile=ct'
```

Note: the Application Image uses an (overrideable) environment variable to activate this.

You might also create your own profiles and use these, please refer to the upstream documentation linked above.

6.5.23 Database Settings

These settings are stored in the `setting` database table but we recommend using the Settings Admin API (*Manage Database Settings*) to view and modify them, as shown below. If changed in the database directly, you need to reload the application to make the ORM pickup the changes.

In short:

- HTTP `GET` is used to show settings.
- HTTP `PUT` is used to add or modify settings.
- HTTP `DELETE` is used to delete settings.

:BlockedApiPolicy (Deprecated)

> **Note**
>
> This setting is deprecated. Please use the JvmSetting *dataverse.api.blocked.policy* instead. This legacy setting will only be used if the newer JvmSettings are not set.

`:BlockedApiPolicy` affects access to the list of API endpoints defined in *:BlockedApiEndpoints (Deprecated)*.

Out of the box, `localhost-only` is the default policy, as mentioned in *Blocking API Endpoints*. The other valid options are the following.

- localhost-only: Allow from localhost.
- unblock-key: Require a key defined in *:BlockedApiKey (Deprecated)*.
- drop: Disallow the blocked endpoints completely.

Below is an example of setting `localhost-only`.

```
curl -X PUT -d localhost-only http://localhost:8080/api/admin/settings/:BlockedApiPolicy
```

:BlockedApiEndpoints (Deprecated)

> **Note**
>
> This setting is deprecated. Please use the JvmSetting *dataverse.api.blocked.endpoints* instead. This legacy setting will only be used if the newer JvmSettings are not set.

A comma-separated list of API endpoints to be blocked. For a standard production installation, the installer blocks both "admin" and "builtin-users" by default per the security section above:

```
curl -X PUT -d "admin,builtin-users" http://localhost:8080/api/admin/settings/
:BlockedApiEndpoints
```

See the *Lists of Dataverse APIs* for lists of API endpoints.

:BlockedApiKey (Deprecated)

> **Note**
>
> This setting is deprecated. Please use the JvmSetting *dataverse.api.blocked.key* instead. This legacy setting will only be used if the newer JvmSettings are not set.

`:BlockedApiKey` is used in conjunction with *:BlockedApiEndpoints (Deprecated)* and *:BlockedApiPolicy (Deprecated)* and will not be enabled unless the policy is set to `unblock-key` as demonstrated below. Please note that the order is significant. You should set `:BlockedApiKey` first to prevent locking yourself out.

```
curl -X PUT -d s3kretKey http://localhost:8080/api/admin/settings/:BlockedApiKey
```

```
curl -X PUT -d unblock-key http://localhost:8080/api/admin/settings/:BlockedApiPolicy
```

Now that `:BlockedApiKey` has been enabled, blocked APIs can be accessed using the header `X-Dataverse-unblock-key: theKeyYouChoose` or, less securely, the query parameter `unblock-key=theKeyYouChose` as in the examples below.

```
curl -H 'X-Dataverse-unblock-key:theKeyYouChoose' https://demo.dataverse.org/api/admin/
settings
```

```
curl https://demo.dataverse.org/api/admin/settings?unblock-key=theKeyYouChose
```

:BuiltinUsersKey

The key required to create users via API as documented at *Native API*. Unlike other database settings, this one doesn't start with a colon.

```
curl -X PUT -d builtInS3kretKey http://localhost:8080/api/admin/settings/:BuiltinUsersKey
```

Note: this key used to be named `BuiltinUsers.KEY` until Dataverse 6.8.

:SearchApiRequiresToken

In Dataverse Software 4.7 and lower, the *Search API* required an API token, but as of Dataverse Software 4.7.1 this is no longer the case. If you prefer the old behavior of requiring API tokens to use the Search API, set `:SearchApiRequiresToken` to `true`.

```
curl -X PUT -d true http://localhost:8080/api/admin/settings/:SearchApiRequiresToken
```

:SystemEmail

Please note that this setting is deprecated since Dataverse 6.2. It will be picked up for backward compatibility, but please migrate to usage of *dataverse.mail.system-email*.

:HomePageCustomizationFile

See *Branding Your Installation* above.

:LogoCustomizationFile

See *Branding Your Installation* above.

:HeaderCustomizationFile

See *Branding Your Installation* above.

:DisableRootDataverseTheme

See *Branding Your Installation* above.

:FooterCustomizationFile

See *Branding Your Installation* above.

:StyleCustomizationFile

See *Branding Your Installation* above.

:WebAnalyticsCode

See *Web Analytics Code* above.

:FooterCopyright

By default the footer says "Copyright © [YYYY]" but you can add text after the year, as in the example below.

```
curl -X PUT -d ", Your Institution" http://localhost:8080/api/admin/settings/
:FooterCopyright
```

Legacy Single PID Provider: :DoiProvider

As of this writing "DataCite" and "EZID" are the only valid options for production installations. Developers using Dataverse Software 4.10+ are welcome to use the keyword "FAKE" to configure a non-production installation with an non-resolving, in-code provider, which will basically short-circuit the DOI publishing process. `:DoiProvider` is only needed if you are using DOI.

```
curl -X PUT -d DataCite http://localhost:8080/api/admin/settings/:DoiProvider
```

This setting relates to the `:Protocol`, `:Authority`, `:Shoulder`, and `:IdentifierGenerationStyle` database settings below as well as the following JVM options:

- *Legacy Single PID Provider: dataverse.pid.datacite.mds-api-url*
- *Legacy Single PID Provider: dataverse.pid.datacite.rest-api-url*

- *Legacy Single PID Provider: dataverse.pid.datacite.username*
- *Legacy Single PID Provider: dataverse.pid.datacite.password*

Legacy Single PID Provider: :Protocol

As of this writing "doi","hdl", and "perma" are the only valid option for the protocol for a persistent ID.

```
curl -X PUT -d doi http://localhost:8080/api/admin/settings/:Protocol
```

Legacy Single PID Provider: :Authority

Use the authority assigned to you by your DoiProvider or HandleProvider, or your choice if using PermaLinks.

Please note that a DOI or Handle authority cannot have a slash ("/") in it (slash is also not recommended for PermaLink authorities).

```
curl -X PUT -d 10.xxxx http://localhost:8080/api/admin/settings/:Authority
```

Legacy Single PID Provider: :Shoulder

The shoulder is used with DOIs and PermaLinks. Out of the box, the shoulder is set to "FK2/" but this is for testing only! When you apply for your DOI authority/namespace, you may have been assigned a shoulder. The following is only an example and a trailing slash is optional.

```
curl -X PUT -d "MyShoulder/" http://localhost:8080/api/admin/settings/:Shoulder
```

Legacy Single PID Provider: :IdentifierGenerationStyle

By default, the Dataverse Software generates a random 6 character string, pre-pended by the Shoulder if set, to use as the identifier for a Dataset. Set this to `storedProcGenerated` to generate instead a custom *unique* identifier (again pre-pended by the Shoulder if set) through a database stored procedure or function (the assumed default setting is `randomString`). In addition to this setting, a stored procedure or function must be created in the database.

As a first example, the script below (downloadable `here`) produces sequential numerical values. You may need to make some changes to suit your system setup, see the comments for more information:

```
-- A script for creating a numeric identifier sequence, and an external
-- stored procedure, for accessing the sequence from inside the application,
-- in a non-hacky, JPA way.

-- NOTE:

-- 1. The database user name "dvnapp" is hard-coded here - it may
-- need to be changed to match your database user name;

-- 2. In the code below, the sequence starts with 1, but it can be adjusted by
-- changing the MINVALUE as needed.

CREATE SEQUENCE datasetidentifier_seq
  INCREMENT 1
  MINVALUE 1
  MAXVALUE 9223372036854775807
  START 1
```

(continues on next page)

(continued from previous page)

```
CACHE 1;

ALTER TABLE datasetidentifier_seq OWNER TO "dvnapp";

-- And now create a PostgreSQL FUNCTION, for JPA to
-- access as a NamedStoredProcedure:

CREATE OR REPLACE FUNCTION generateIdentifierFromStoredProcedure()
RETURNS varchar AS $$
DECLARE
  identifier varchar;
BEGIN
  identifier := nextval('datasetidentifier_seq')::varchar;
  RETURN identifier;
END;
$$ LANGUAGE plpgsql IMMUTABLE;
```

As a second example, the script below (downloadable here) produces sequential 8 character identifiers from a base36 representation of current timestamp.

```
-- A script for creating, through a database stored procedure, sequential
-- 8 character identifiers from a base36 representation of current timestamp.

CREATE OR REPLACE FUNCTION base36_encode(
  IN digits bigint, IN min_width int = 0)
RETURNS varchar AS $$
DECLARE
    chars char[];
    ret varchar;
    val bigint;
BEGIN
    chars := ARRAY[
      '0','1','2','3','4','5','6','7','8','9',
      'a','b','c','d','e','f','g','h','i','j',
      'k','l','m','n','o','p','q','r','s','t',
      'u','v','w','x','y','z'];
    val := digits;
    ret := '';
    IF val < 0 THEN
        val := val * -1;
    END IF;
    WHILE val != 0 LOOP
        ret := chars[(val % 36)+1] || ret;
        val := val / 36;
    END LOOP;

    IF min_width > 0 AND char_length(ret) < min_width THEN
        ret := lpad(ret, min_width, '0');
    END IF;

    RETURN ret;
END;
```

(continues on next page)

(continued from previous page)

```
$$ LANGUAGE plpgsql IMMUTABLE;


CREATE OR REPLACE FUNCTION generateIdentifierFromStoredProcedure()
RETURNS varchar AS $$
DECLARE
    curr_time_msec bigint;
    identifier varchar;
BEGIN
    curr_time_msec := extract(epoch from now())*1000;
    identifier := base36_encode(curr_time_msec);
    RETURN identifier;
END;
$$ LANGUAGE plpgsql IMMUTABLE;
```

Note that the SQL in these examples scripts is Postgres-specific. If necessary, it can be reimplemented in any other SQL flavor - the standard JPA code in the application simply expects the database to have a saved function ("stored procedure") named `generateIdentifierFromStoredProcedure()` returning a single `varchar` argument.

Please note that `:IdentifierGenerationStyle` also plays a role for the "identifier" for files. See the section on *Legacy Single PID Provider: :DataFilePIDFormat* below for more details.

Legacy Single PID Provider: :DataFilePIDFormat

This setting controls the way that the "identifier" component of a file's persistent identifier (PID) relates to the PID of its "parent" dataset.

By default the identifier for a file is dependent on its parent dataset. For example, if the identifier of a dataset is "TJCLKP", the identifier for a file within that dataset will consist of the parent dataset's identifier followed by a slash ("/"), followed by a random 6 character string, yielding "TJCLKP/MLGWJO". Identifiers in this format are what you should expect if you leave `:DataFilePIDFormat` undefined or set it to `DEPENDENT` and have not changed the `:IdentifierGenerationStyle` setting from its default.

Alternatively, the identifier for File PIDs can be configured to be independent of Dataset PIDs using the setting `INDEPENDENT`. In this case, file PIDs will not contain the PIDs of their parent datasets, and their PIDs will be generated the exact same way that datasets' PIDs are, based on the `:IdentifierGenerationStyle` setting described above (random 6 character strings or custom unique identifiers through a stored procedure, pre-pended by any shoulder).

The chart below shows examples from each possible combination of parameters from the two settings. `:IdentifierGenerationStyle` can be either `randomString` (the default) or `storedProcGenerated` and `:DataFilePIDFormat` can be either `DEPENDENT` (the default) or `INDEPENDENT`. In the examples below the "identifier" for the dataset is "TJCLKP" for `randomString` and "100001" for `storedProcGenerated` (when using sequential numerical values, as described in *Legacy Single PID Provider: :IdentifierGenerationStyle* above), or "krby26qt" for `storedProcGenerated` (when using base36 timestamps, as described in *Legacy Single PID Provider: :IdentifierGenerationStyle* above).

	randomString	storedProcGenerated (sequential numbers)	storedProcGenerated (base36 timestamps)
DEPENDENT	TJCLKP/MLGWJC	100001/1	krby26qt/1
INDEPENDENT	MLGWJO	100002	krby27pz

As seen above, in cases where `:IdentifierGenerationStyle` is set to `storedProcGenerated` and `:DataFilePIDFormat` is set to `DEPENDENT`, each file within a dataset will be assigned a number *within* that dataset starting with "1".

Otherwise, if `:DataFilePIDFormat` is set to `INDEPENDENT`, each file within the dataset is assigned with a new PID which is the next available identifier provided from the database stored procedure. In our example: "100002" when using sequential numbers or "krby27pz" when using base36 timestamps.

:FilePIDsEnabled

Toggles publishing of file-level PIDs for the entire installation. By default this setting is absent and Dataverse Software assumes it to be false. If enabled, the registration will be performed asynchronously (in the background) during publishing of a dataset.

It is possible to override the installation-wide setting for specific collections, see *:AllowEnablingFilePIDsPerCollection*. For example, registration of PIDs for files can be enabled in a specific collection when it is disabled instance-wide. Or it can be disabled in specific collections where it is enabled by default. See *Change Collection Attributes* for details.

To enable file-level PIDs for the entire installation:

```
``curl -X PUT -d 'true' http://localhost:8080/api/admin/settings/:FilePIDsEnabled``
```

If you don't want to register file-based PIDs for your entire installation:

```
``curl -X PUT -d 'false' http://localhost:8080/api/admin/settings/:FilePIDsEnabled``
```

:AllowEnablingFilePIDsPerCollection

Toggles whether superusers can change the File PIDs policy per collection. By default this setting is absent and Dataverse Software assumes it to be false.

For example, if this setting is true, registration of PIDs for files can be enabled in a specific collection when it is disabled instance-wide. Or it can be disabled in specific collections where it is enabled by default. See *Change Collection Attributes* for details.

To enable setting file-level PIDs per collection:

```
``curl -X PUT -d 'true' http://localhost:8080/api/admin/settings/
↪:AllowEnablingFilePIDsPerCollection``
```

When :AllowEnablingFilePIDsPerCollection is true, setting File PIDs to be enabled/disabled for a given collection can be done via the Native API - see *Change Collection Attributes* in the Native API Guide.

Legacy Single PID Provider: :IndependentHandleService

Specific for Handle PIDs. Set this setting to true if you want to use a Handle service which is setup to work 'independently' (No communication with the Global Handle Registry). By default this setting is absent and the Dataverse Software assumes it to be false.

```
curl -X PUT -d 'true' http://localhost:8080/api/admin/settings/:IndependentHandleService
```

Legacy Single PID Provider: :HandleAuthHandle

Specific for Handle PIDs. Set this setting to <prefix>/<suffix> to be used on a global handle service when the public key is NOT stored in the default handle. By default this setting is absent and the Dataverse Software assumes it to be not set. If the public key for instance is stored in handle: 21.T12996/USER01. For this handle the prefix is '21.T12996' and the suffix is 'USER01'. The command to execute is then:

```
curl -X PUT -d '21.T12996/USER01' http://localhost:8080/api/admin/settings/
:HandleAuthHandle
```

:FileValidationOnPublishEnabled

Toggles validation of the physical files in the dataset when it's published, by recalculating the checksums and comparing against the values stored in the DataFile table. By default this setting is absent and the Dataverse Software assumes it to be true. If enabled, the validation will be performed asynchronously, similarly to how we handle assigning persistent identifiers to datafiles, with the dataset locked for the duration of the publishing process.

If you don't want the datafiles to be validated on publish, set:

```
curl -X PUT -d 'false' http://localhost:8080/api/admin/settings/:FileValidationOnPublishEnabled
```

:ApplicationTermsOfUse

Application Terms of Use (called "General Terms of Use" in the UI) are shown to the user when they sign up for an account. Some HTML tags are supported. For a list, see *Supported HTML Tags* in the User Guide.

You can set terms like this:

```
curl -X PUT http://localhost:8080/api/admin/settings/:ApplicationTermsOfUse --upload-file
/tmp/apptou.html
```

To delete terms:

```
curl -X DELETE http://localhost:8080/api/admin/settings/:ApplicationTermsOfUse
```

If *Internationalization* is enabled, you can set the terms per language. To set terms for a specific language ("fr", for example):

```
curl -X PUT http://localhost:8080/api/admin/settings/:ApplicationTermsOfUse/lang/fr
--upload-file /tmp/apptou_fr.html
```

To delete terms for a specific language ("fr", for example):

```
curl -X DELETE http://localhost:8080/api/admin/settings/:ApplicationTermsOfUse/lang/fr
```

To retrieve the values, see *Get Application Terms of Use (General Terms of Use)* in the API Guide.

:ApplicationPrivacyPolicyUrl

Specify a URL where users can read your Privacy Policy, linked from the bottom of the page.

```
curl -X PUT -d https://dataverse.org/best-practices/harvard-dataverse-privacy-policy
http://localhost:8080/api/admin/settings/:ApplicationPrivacyPolicyUrl
```

:ApiTermsOfUse

Specify a URL where users can read your API Terms of Use. API users can retrieve this URL from the SWORD Service Document or the *Info* section of our *Native API* documentation.

```
curl -X PUT -d https://dataverse.org/best-practices/harvard-api-tou http://
localhost:8080/api/admin/settings/:ApiTermsOfUse
```

:ExcludeEmailFromExport

See also *Privacy Considerations* above.

Set `:ExcludeEmailFromExport` to prevent email addresses for contacts from being exposed in XML or JSON representations of dataset and Dataverse collection metadata. For a list exported formats such as DDI, see the *Metadata Export* section of the Admin Guide.

```
curl -X PUT -d true http://localhost:8080/api/admin/settings/:ExcludeEmailFromExport
```

Note: After making a change to this setting, a reExportAll needs to be run before the changes will be reflected in the exports:

```
curl http://localhost:8080/api/admin/metadata/reExportAll
```

This will *force* a re-export of every published, local dataset, regardless of whether it has already been exported or not.

The call returns a status message informing the administrator, that the process has been launched (`{"status":"WORKFLOW_IN_PROGRESS"}`). The administrator can check the progress of the process via log files: `[Payara directory]/glassfish/domains/domain1/logs/export_[time stamp].log`.

:NavbarAboutUrl

Set `:NavbarAboutUrl` to a fully-qualified URL which will be used for the "About" link in the navbar.

Note: The "About" link will not appear in the navbar until this option is set.

```
curl -X PUT -d http://dataverse.example.edu http://localhost:8080/api/admin/settings/
:NavbarAboutUrl
```

:NavbarGuidesUrl

Set `:NavbarGuidesUrl` to a fully-qualified URL which will be used for the "User Guide" link in the navbar.

Note: by default, the URL is composed from the settings `:GuidesBaseUrl` and `:GuidesVersion` below.

```
curl -X PUT -d http://example.edu/fancy-dataverse-guide http://localhost:8080/api/admin/
settings/:NavbarGuidesUrl
```

:GuidesBaseUrl

Set `:GuidesBaseUrl` to override the default value "https://guides.dataverse.org". If you are interested in writing your own version of the guides, you may find the *Writing Documentation* section of the Contributor Guide helpful.

```
curl -X PUT -d http://dataverse.example.edu http://localhost:8080/api/admin/settings/
:GuidesBaseUrl
```

:GuidesVersion

Set `:GuidesVersion` to override the version number in the URL of guides. For example, rather than http://guides.dataverse.org/en/4.6/user/account.html the version is overriden to http://guides.dataverse.org/en/1234-new-feature/user/account.html in the example below:

```
curl -X PUT -d 1234-new-feature http://localhost:8080/api/admin/settings/:GuidesVersion
```

:NavbarSupportUrl

Set `:NavbarSupportUrl` to a fully-qualified URL which will be used for the "Support" link in the navbar.

Note that this will override the default behaviour for the "Support" menu option, which is to display the Dataverse collection 'feedback' dialog.

```
curl -X PUT -d https://dataverse.example.edu/supportpage.html http://localhost:8080/api/
admin/settings/:NavbarSupportUrl
```

:MetricsUrl

Make the metrics component on the root Dataverse collection a clickable link to a website where you present metrics on your Dataverse installation, perhaps one of the community-supported tools mentioned in the *Reporting Tools and Common Queries* section of the Admin Guide.

```
curl -X PUT -d https://metrics.dataverse.example.edu http://localhost:8080/api/admin/
settings/:MetricsUrl
```

:MaxFileUploadSizeInBytes

This setting controls the maximum size of uploaded files. - To have one limit for all stores, set *MaxFileUploadSizeInBytes* to "2147483648", for example, to limit the size of files uploaded to 2 GB:

```
curl -X PUT -d 2147483648 http://localhost:8080/api/admin/settings/
:MaxFileUploadSizeInBytes
```

- To have limits per store with an optional default, use a serialized json object for the value of *MaxFileUploadSizeInBytes* with an entry per store, as in the following example, which maintains a 2 GB default and adds higher limits for stores with ids "fileOne" and "s3".

```
curl -X PUT -d '{"default":"2147483648","fileOne":"4000000000","s3":"8000000000"}' http:/
/localhost:8080/api/admin/settings/:MaxFileUploadSizeInBytes
```

Notes:

- For SWORD, this size is limited by the Java Integer.MAX_VALUE of 2,147,483,647. (see: https://github.com/IQSS/dataverse/issues/2169)
- If the MaxFileUploadSizeInBytes is NOT set, uploads, including SWORD may be of unlimited size.

- For larger file upload sizes, you may need to configure your reverse proxy timeout. If using apache2 (httpd) with Shibboleth, add a timeout to the ProxyPass defined in etc/httpd/conf.d/ssl.conf (which is described in the *Shibboleth* setup).

:MultipleUploadFilesLimit

This setting controls the number of files that can be uploaded through the UI at once. The default is 1000. It should be set to 1 or higher since 0 has no effect. To limit the number of files in a zip file, see `:ZipUploadFilesLimit`.

```
curl -X PUT -d 500 http://localhost:8080/api/admin/settings/:MultipleUploadFilesLimit
```

:ZipDownloadLimit

For performance reasons, your Dataverse installation will only allow creation of zip files up to 100 MB, but the limit can be increased. Here's an example of raising the limit to 1 GB:

```
curl -X PUT -d 1000000000 http://localhost:8080/api/admin/settings/:ZipDownloadLimit
```

In the UI, users trying to download a zip file larger than the Dataverse installation's :ZipDownloadLimit will receive messaging that the zip file is too large, and the user will be presented with alternate access options.

:TabularIngestSizeLimit

Threshold in bytes for limiting whether or not "ingest" is attempted for an uploaded tabular file (which can be resource intensive). For more on the ingest feature, see *Tabular Data File Ingest* in the User Guide.

There are two ways to specify ingest size limits. You can set a global limit for all file types or you can use a JSON file for more granularity. We'll cover the global limit first.

With the following value in place (again, expressed in bytes), files greater than 2 GB in size will not go through the ingest process:

```
curl -X PUT -d 2000000000 http://localhost:8080/api/admin/settings/
:TabularIngestSizeLimit
```

You can set this value to `0` to prevent files from being ingested at all.

Out of the box, the `:TabularIngestSizeLimit` setting is absent, which results in ingest being attempted no matter how large the file is. You can specify this "no size limit" default explicitly with the value `-1`.

Using a JSON-based setting, you can set a global default and per-format limits for the following formats:

- CSV
- DTA
- POR
- Rdata
- SAV
- XLSX

(In previous releases of Dataverse, a colon-separated form was used to specify per-format limits, such as `:TabularIngestSizeLimit:Rdata`, but this is no longer supported. Now JSON is used.)

The expected JSON is an object with key/value pairs like the following. Format names are case-insensitive, and all fields are optional (an empty JSON object equals not restricted). The size limits must be whole numbers, either presented as

strings with double quotes around them (e.g. "10") or numeric values (e.g. 10 or 10.0). Note that decimal numbers like 10.5 are invalid. Any invalid setting will temporarily disable tabular ingest until corrected.

```
{
  "default": "-1",
  "csv": "0",
  "dta": "10",
  "por": "100"
}
```

Whatever JSON you send will overwrite existing values. If you have any exiting :TabularIngestSizeLimit settings, you can use the following command to see them in the expected input format above (and then add the new settings you want):

```
curl http://localhost:8080/api/admin/settings/:TabularIngestSizeLimit | jq -r '.data.
message'
```

The default key is optional and can be used to give limits to formats that are not specified in the JSON. If you omit the default key or set it to "-1", no limits are applied to formats not specified in the JSON. If you set it to "0", ingest will be disabled (but you can override this per-format). If you wish to disable ingest for specific formats it may be preferable to instead set a value of "1" which will cause Dataverse to print a string such as rdata:disabled for the user's benefit.

Add a format name (csv, dta, etc., as listed above) to change the limit for that particular format.

Examples:

1. If you want your Dataverse installation to not attempt to ingest Rdata files larger than 1 MB but otherwise be unlimited:

   ```
   curl -X PUT -d '{"Rdata":"1000000"}' http://localhost:8080/api/admin/settings/
   :TabularIngestSizeLimit
   ```

2. If you want your Dataverse installation to not attempt to ingest XLSX files at all and apply a global limit of 512 MiB, use this setting:

   ```
   curl -X PUT -d '{"default":"536870912", "XSLX":"1"}' http://localhost:8080/api/
   admin/settings/:TabularIngestSizeLimit
   ```

3. If you want your Dataverse installation to not attempt to ingest files at all except for CSV files that are 256 MiB or smaller, use this setting:

   ```
   curl -X PUT -d '{"default":"1", "CSV":"268435456"}' http://localhost:8080/api/admin/
   settings/:TabularIngestSizeLimit
   ```

:HarvestingClientCallRateLimit

This setting allows configuring sleep intervals between OAI calls for specific harvesting clients. Which makes it possible to harvest from servers that enforce rate limits.

The setting value is a serialized JSON object mapping client names to the specified intervals in fractional seconds. It is also possible to set a universal default interval for all harvesting clients on the instance (in a somewhat unlikely use case where this may be practically necessary).

In the following example, the harvester is instructed to sleep for 900 milliseconds between calls when running the client named harvarddv, and to default to zero otherwise:

```
curl -X PUT -d "{\"harvarddv\":  0.9, \"default\":  0}" "http://localhost:8080/api/admin/
settings/:HarvestingClientCallRateLimit"
```

Please note that the default in the example above is there for illustrative purposes and is otherwise redundant, since no sleep interval is the default behavior anyway.

:ZipUploadFilesLimit

Limit the number of files in a zip that your Dataverse installation will accept. In the absence of this setting, your Dataverse instalation defaults to a limit of 1,000 files per zipfile.

```
curl -X PUT -d 2048 http://localhost:8080/api/admin/settings/:ZipUploadFilesLimit
```

:SolrHostColonPort

By default your Dataverse installation will attempt to connect to Solr on port 8983 on localhost. Use this setting to change the hostname or port. You must restart Payara after making this change.

```
curl -X PUT -d localhost:8983 http://localhost:8080/api/admin/settings/:SolrHostColonPort
```

Note: instead of using a database setting, you could alternatively use JVM settings like *dataverse.solr.host*.

:SolrFullTextIndexing

Whether or not to index the content of files such as PDFs. The default is false.

```
curl -X PUT -d true http://localhost:8080/api/admin/settings/:SolrFullTextIndexing
```

:SolrMaxFileSizeForFullTextIndexing

If `:SolrFullTextIndexing` is set to true, the content of files of any size will be indexed. To set a limit in bytes for which files to index in this way:

```
curl -X PUT -d 314572800 http://localhost:8080/api/admin/settings/
:SolrMaxFileSizeForFullTextIndexing
```

:DisableSolrFacets

Setting this to `true` will make the collection ("dataverse") page start showing search results without the usual search facets on the left side of the page. A message will be shown in that column informing the users that facets are temporarily unavailable. Generating the facets may in some cases be more resource-intensive for Solr than the main search results themselves, so applying this measure will significantly reduce the load on the search engine when its performance becomes an issue.

This setting can be used in combination with the "circuit breaker" mechanism on the Solr side (see the "Installing Solr" section of the Installation Prerequisites guide). An admin can choose to enable it, or even create an automated system for enabling it in response to Solr beginning to drop incoming requests with the HTTP code 503.

To enable the setting:

```
curl -X PUT -d true "http://localhost:8080/api/admin/settings/:DisableSolrFacets"
```

:DisableSolrFacetsForGuestUsers

Similar to the above, but will disable the facets for Guest (unauthenticated) users only.

:DisableSolrFacetsWithoutJsession

Same idea as with the 2 settings above. For the purposes of this setting, a request is considered "anonymous", if it came in without the JSESSION cookie supplied. A UI user who is browsing the holdings without logging in will have a valid JSESSION cookie, tied to a guest session. The main purpose of this setting is to hide the facets from bots, scripted crawlers and such (most of which - though not all - do not use cookies). Not letting the bots anywhere near the facets can serve a dual purpose on a busy instance experiencing problems with such abuse - some CPU cycles and resources can be saved by not having to generate the facets. And, even more importantly, it can prevent bots from attempting to crawl the facet trees, which has a potential for multiplying the service load.

:DisableUncheckedTypesFacet

Another option for reducing the load on solr on a busy instance. Rather than disabling all the search facets, this setting affects only one - the facet on the upper left of the collection page, where users can select the type of objects to search - Collections ("Dataverses"), Datasets and/or Files. With this option set to true, the numbers of results will only be shown for the types actually selected (i.e. only for the search results currently shown to the user). This minor feature - being able to tell the user how many files (for example) they *would* find, *if* they chose to search for files, by clicking the Files facet - essentially doubles the expense of running the search. That may still be negligible on an instance with lighter holdings, but can make a significant difference for a large and heavily used archive.

:SignUpUrl

The relative path URL to which users will be sent for sign up. The default setting is below.

```
curl -X PUT -d '/dataverseuser.xhtml?editMode=CREATE' http://localhost:8080/api/admin/
settings/:SignUpUrl
```

:LoginSessionTimeout

Session timeout (in minutes) for logged-in users. The default is 8 hours (480 minutes). For the anonymous user sessions, the timeout is set to the default value, configured in the web.xml file of the Dataverse installation.

In the example below we reduce the timeout to 4 hours:

```
curl -X PUT -d 240 http://localhost:8080/api/admin/settings/:LoginSessionTimeout
```

:DatasetPublishPopupCustomText

Set custom text a user will view when publishing a dataset. Note that this text is exposed via the "Info" endpoint of the *Native API*.

```
curl -X PUT -d "Deposit License Requirements" http://localhost:8080/api/admin/settings/
:DatasetPublishPopupCustomText
```

If you have a long text string, you can upload it as a file as in the example below.

```
curl -X PUT --upload-file /tmp/long.txt http://localhost:8080/api/admin/settings/
:DatasetPublishPopupCustomText
```

There is a related setting called *:PublishDatasetDisclaimerText* that also makes text appear on the popup when publishing, but it requires a checkbox to be clicked.

See also *Show Custom Popup Text for Publishing Datasets* in the API Guide.

:DatasetPublishPopupCustomTextOnAllVersions

Set whether a user will see the custom text when publishing all versions of a dataset

```
curl -X PUT -d true http://localhost:8080/api/admin/settings/:DatasetPublishPopupCustomTextOnAllVersions
```

:DatasetSubmitForReviewPopupCustomText

Set custom text a user will view when submitting a dataset for review. Note that this text is exposed via the "Info" endpoint of the *Native API*.

```
curl -X PUT -d "Deposit License Requirements" http://localhost:8080/api/admin/settings/
:DatasetSubmitForReviewPopupCustomText
```

If you have a long text string, you can upload it as a file as in the example below.

```
curl -X PUT --upload-file /tmp/long.txt http://localhost:8080/api/admin/settings/
:DatasetSubmitForReviewPopupCustomText
```

There is a related setting called *:SubmitForReviewDatasetDisclaimerText* that also makes text appear on the popup when submitting for review, but it requires a checkbox to be clicked.

See also *Show Custom Popup Text for Submitting Datasets For Review* in the API Guide.

:SearchHighlightFragmentSize

Set `:SearchHighlightFragmentSize` to override the default value of 100 from https://wiki.apache.org/solr/HighlightingParameters#hl.fragsize . In practice, a value of "320" seemed to fix the issue at https://github.com/IQSS/dataverse/issues/2191

```
curl -X PUT -d 320 http://localhost:8080/api/admin/settings/:SearchHighlightFragmentSize
```

:ScrubMigrationData

Allow for migration of non-conformant data (especially dates) from Dataverse Software 3.x to Dataverse Software 4.x

:MinutesUntilConfirmEmailTokenExpires

The duration in minutes before "Confirm Email" URLs expire. The default is 1440 minutes (24 hours). See also the *User Administration* section of our Admin Guide.

:DefaultAuthProvider

If you have enabled Shibboleth and/or one or more OAuth providers, you may wish to make one of these authentication providers the default when users visit the Log In page. If unset, this will default to `builtin` but these valid options (depending if you've done the setup described in the *Shibboleth* or *OAuth Login Options* sections) are:

- `builtin`
- `shib`
- `orcid`
- `github`
- `google`

Here is an example of setting the default auth provider back to `builtin`:

```
curl -X PUT -d builtin http://localhost:8080/api/admin/settings/:DefaultAuthProvider
```

:AllowSignUp

Set to false to disallow local accounts from being created. See also the sections on *Shibboleth* and *OAuth Login Options*.

```
curl -X PUT -d 'false' http://localhost:8080/api/admin/settings/:AllowSignUp
```

:AllowRemoteAuthSignUp

This is a **compound** setting that enables or disables sign up for new accounts for individual OAuth2 authentication methods (such as Orcid, Google and GitHub). This way it is possible to continue allowing logins via an OAuth2 provider for already existing accounts, without letting new users create accounts with this method.

By default, if the setting is not present, all configured OAuth sign ups are allowed. If the setting is present, but the value for this specific method is not specified, it is assumed that the sign ups are allowed for it.

Examples:

This curl command…

```
curl -X PUT -d '{"default":"false"}' http://localhost:8080/api/admin/settings/
:AllowRemoteAuthSignUp
```

…disables all OAuth sign ups.

This curl command…

```
curl -X PUT -d '{"default":"true","google":"false"}' http://localhost:8080/api/admin/
settings/:AllowRemoteAuthSignUp
```

…keeps sign ups open for all the OAuth login providers except google. (That said, note that the `"default":"true"` part in this example is redundant, since it would default to true anyway for all the methods other than google.)

See also *OAuth Login Options*.

:FileFixityChecksumAlgorithm

The Dataverse Software calculates checksums for uploaded files so that users can determine if their file was corrupted via upload or download. This is sometimes called "file fixity": https://en.wikipedia.org/wiki/File_Fixity

The default checksum algorithm used is MD5 and should be sufficient for establishing file fixity. "SHA-1", "SHA-256" and "SHA-512" are alternate values for this setting. For example:

```
curl -X PUT -d 'SHA-512' http://localhost:8080/api/admin/settings/
:FileFixityChecksumAlgorithm
```

To update the algorithm used for existing files, see *Update Checksums To Use New Algorithm*

The fixity checksum algorithm in use can be discovered via API. See *Get Fixity Algorithm* in the API Guide.

:PVMinLength

Password policy setting for builtin user accounts: a password's minimum valid character length. The default is 6.

```
curl -X PUT -d 6 http://localhost:8080/api/admin/settings/:PVMinLength
```

:PVMaxLength

Password policy setting for builtin user accounts: a password's maximum valid character length.

```
curl -X PUT -d 0 http://localhost:8080/api/admin/settings/:PVMaxLength
```

:PVNumberOfConsecutiveDigitsAllowed

By default, passwords can contain an unlimited number of digits in a row. However, if your password policy specifies otherwise (e.g. only four digits in a row are allowed), then you can issue the following curl command to set the number of consecutive digits allowed (this example uses 4):

```
curl -X PUT -d 4 http://localhost:8080/api/admin/settings/:PVNumberOfConsecutiveDigitsAllowed
```

:PVCharacterRules

Password policy setting for builtinuser accounts: dictates which types of characters can be required in a password. This setting goes hand-in-hand with *:PVNumberOfCharacteristics*. The default setting contains two rules:

- one letter
- one digit

The default setting above is equivalent to specifying "Alphabetical:1,Digit:1".

By specifying "UpperCase:1,LowerCase:1,Digit:1,Special:1", for example, you can put the following four rules in place instead:

- one uppercase letter
- one lowercase letter
- one digit
- one special character

If you have implemented 4 different character rules in this way, you can also optionally increase `:PVNumberOfCharacteristics` to as high as 4. However, please note that `:PVNumberOfCharacteristics` cannot be set to a number higher than the number of character rules or you will see the error, "Number of characteristics must be <= to the number of rules".

Also note that the Alphabetical setting should not be used in tandem with the UpperCase or LowerCase settings. The Alphabetical setting encompasses both of those more specific settings, so using it with them will cause your password policy to be unnecessarily confusing, and potentially easier to bypass.

```
curl -X PUT -d 'UpperCase:1,LowerCase:1,Digit:1,Special:1' http://localhost:8080/api/
admin/settings/:PVCharacterRules
```

```
curl -X PUT -d 3 http://localhost:8080/api/admin/settings/:PVNumberOfCharacteristics
```

:PVNumberOfCharacteristics

Password policy setting for builtin user accounts: the number indicates how many of the character rules defined by `:PVCharacterRules` are required as part of a password. The default is 2. `:PVNumberOfCharacteristics` cannot be set to a number higher than the number of rules or you will see the error, "Number of characteristics must be <= to the number of rules".

```
curl -X PUT -d 2 http://localhost:8080/api/admin/settings/:PVNumberOfCharacteristics
```

:PVDictionaries

Password policy setting for builtin user accounts: set a comma separated list of dictionaries containing words that cannot be used in a user password. `/usr/share/dict/words` is suggested and shown modified below to not contain words 3 letters or less. You are free to choose a different dictionary. By default, no dictionary is checked.

```
DIR=THE_PATH_YOU_WANT_YOUR_DICTIONARY_TO_RESIDE  sed '/^.\{,3\}$/d' /usr/share/dict/words
> $DIR/pwdictionary   curl -X PUT -d "$DIR/pwdictionary" http://localhost:8080/api/admin/
settings/:PVDictionaries
```

:PVGoodStrength

Password policy setting for builtin user accounts: passwords of equal or greater character length than the :PVGood-Strength setting are always valid, regardless of other password constraints.

```
curl -X PUT -d 20 http://localhost:8080/api/admin/settings/:PVGoodStrength
```

Recommended setting: 20.

:PVCustomPasswordResetAlertMessage

Changes the default info message displayed when a user is required to change their password on login. The default is:

```
{0} Reset Password{1} - Our password requirements have changed. Please pick a strong
password that matches the criteria below.
```

Where the {0} and {1} denote surrounding HTML **bold** tags. It's recommended to put a single space before your custom message for better appearance (as in the default message above). Including the {0} and {1} to bolden part of your message is optional.

Customize the message using the following curl command's syntax:

```
curl -X PUT -d '{0} Action Required:{1} Your current password does not meet all
requirements. Please enter a new password meeting the criteria below.' http://
localhost:8080/api/admin/settings/:PVCustomPasswordResetAlertMessage
```

:ShibPassiveLoginEnabled

Set `:ShibPassiveLoginEnabled` to true to enable passive login for Shibboleth. When this feature is enabled, an additional Javascript file (isPassive.js) will be loaded for every page. It will generate a passive login request to your Shibboleth SP when an anonymous user navigates to the site. A cookie named "_check_is_passive_dv" will be created to keep track of whether or not a passive login request has already been made for the user.

This implementation follows the example on the Shibboleth wiki documentation page for the isPassive feature: https://wiki.shibboleth.net/confluence/display/SHIB2/isPassive

It is recommended that you configure additional error handling for your Service Provider if you enable passive login. A good way of doing this is described in the Shibboleth wiki documentation:

- *In your Service Provider 2.x shibboleth2.xml file, add redirectErrors="#THIS PAGE#" to the Errors element.*

You can set the value of "#THIS PAGE#" to the URL of your Dataverse installation homepage, or any other page on your site that is accessible to anonymous users and will have the isPassive.js file loaded.

```
curl -X PUT -d true http://localhost:8080/api/admin/settings/:ShibPassiveLoginEnabled
```

:ShibAffiliationAttribute

The Shibboleth affiliation attribute holds information about the affiliation of the user (e.g. "OU") and is read from the DiscoFeed at each login. `:ShibAffiliationAttribute` is a name of a Shibboleth attribute in the Shibboleth header which your Dataverse installation will read from instead of DiscoFeed. If this value is not set or empty, your Dataverse installation uses the DiscoFeed.

If the attribute is not yet set for the Shibboleth, please consult the Shibboleth Administrators at your institution. Typically it requires changing of the */etc/shibboleth/attribute-map.xml* file by adding an attribute request, e.g.

```
<Attribute name="urn:oid:2.5.4.11" id="ou">
    <AttributeDecoder xsi:type="StringAttributeDecoder" caseSensitive="false"/>
</Attribute>
```

In order to implement the change, you should restart Shibboleth and Apache2 services:

```
sudo service shibd restart
sudo service apache2 restart
```

To check if the attribute is sent, you should log in again to the Dataverse installation and check Shibboleth's transaction log. You should see something like this:

```
INFO Shibboleth-TRANSACTION [25]: Cached the following attributes with session (ID: _
↪9d1f34c0733b61c0feb0ca7596ef43b2) for (applicationId: default) {
INFO Shibboleth-TRANSACTION [25]:       givenName (1 values)
INFO Shibboleth-TRANSACTION [25]:       ou (1 values)
INFO Shibboleth-TRANSACTION [25]:       sn (1 values)
INFO Shibboleth-TRANSACTION [25]:       eppn (1 values)
INFO Shibboleth-TRANSACTION [25]:       mail (1 values)
INFO Shibboleth-TRANSACTION [25]:       displayName (1 values)
INFO Shibboleth-TRANSACTION [25]: }
```

If you see the attribue you requested in this list, you can set the attribute in the Dataverse installation.

To set `:ShibAffiliationAttribute`:

```
curl -X PUT -d "ou" http://localhost:8080/api/admin/settings/:ShibAffiliationAttribute
```

To delete `:ShibAffiliationAttribute`:

```
curl -X DELETE http://localhost:8080/api/admin/settings/:ShibAffiliationAttribute
```

To check the current value of `:ShibAffiliationAttribute`:

```
curl -X GET http://localhost:8080/api/admin/settings/:ShibAffiliationAttribute
```

:ShibAttributeCharacterSetConversionEnabled

It seems that the application server (usually Glassfish or Payara) will interpret all Shibboleth attributes that come through AJP as ISO-8859-1, even if they where originally UTF-8. To circumvent that, we re-encode all received Shibboleth attributes manually as UTF-8 by default. In the case you get garbled characters in Shibboleth-supplied fields (e.g. given name, surname, affiliation), you can disable this behaviour by setting ShibAttributeCharacterSetConversionEnabled to false:

```
curl -X PUT -d false http://localhost:8080/api/admin/settings/:ShibAttributeCharacterSetConversionEnable
```

If you managed to get correct accented characters from shibboleth while this setting is _false_, please contact us with your application server and Shibboleth configuration!

:ShibAffiliationOrder

Will select the last or first value of an array in affiliation, the array separator can be set using `:ShibAffiliationSeparator` .

To select the last value :

```
curl -X PUT -d "lastAffiliation" http://localhost:8080/api/admin/settings/
:ShibAffiliationOrder
```

To select the first value :

```
curl -X PUT -d "firstAffiliation" http://localhost:8080/api/admin/settings/
:ShibAffiliationOrder
```

:ShibAffiliationSeparator

Set the separator to be used for `:ShibAffiliationOrder`. Default separator : ";"

To change the separator :

```
curl -X PUT -d ";" http://localhost:8080/api/admin/settings/:ShibAffiliationSeparator
```

:ComputeBaseUrl

Set the base URL for the "Compute" button for a dataset.

```
curl -X PUT -d 'https://giji.massopencloud.org/application/dataverse' http://
localhost:8080/api/admin/settings/:ComputeBaseUrl
```

:CloudEnvironmentName

Set the name of the cloud environment you've integrated with your Dataverse installation.

```
curl -X PUT -d 'Massachusetts Open Cloud (MOC)' http://localhost:8080/api/admin/settings/
:CloudEnvironmentName
```

:PublicInstall

Setting an installation to public will remove the ability to restrict data files or datasets. This functionality of the Dataverse Software will be disabled from your installation.

This is useful for specific cases where an installation's files are stored in public access. Because files stored this way do not obey the Dataverse Software's file restrictions, users would still be able to access the files even when they're restricted. In these cases it's best to use :PublicInstall to disable the feature altogether.

```
curl -X PUT -d true http://localhost:8080/api/admin/settings/:PublicInstall
```

:UploadMethods

This setting controls which upload methods are available to users of your Dataverse installation. The following upload methods are available:

- `native/http`: Corresponds to "Upload with HTTP via your browser" and APIs that use HTTP (SWORD and native).
- `dvwebloader`: Corresponds to *Folder Upload*. Note that `dataverse.files.<id>.upload-redirect` must be set to "true" on an S3 store for this method to show up in the UI. In addition, *:WebloaderUrl* must be set. CORS allowed on the S3 bucket. See *Allow CORS for S3 Buckets*.

Out of the box only `native/http` is enabled and will work without further configuration. To add multiple upload method, separate them using a comma like this:

```
curl -X PUT -d 'native/http,dvwebloader' http://localhost:8080/api/admin/settings/
:UploadMethods
```

You'll always want at least one upload method, so the easiest way to remove one of them is to simply `PUT` just the one you want, like this:

```
curl -X PUT -d 'native/http' http://localhost:8080/api/admin/settings/:UploadMethods
```

:GuestbookResponsesPageDisplayLimit

Limit on how many guestbook entries to display on the guestbook-responses page. By default, only the 5000 most recent entries will be shown. Use the standard settings API in order to change the limit. For example, to set it to 10,000, make the following API call:

```
curl -X PUT -d 10000 http://localhost:8080/api/admin/settings/:GuestbookResponsesPageDisplayLimit
```

:CustomDatasetSummaryFields

You can replace the default dataset metadata fields that are displayed above files table on the dataset page with a custom list separated by commas (with optional spaces) using the curl command below. Note that the License is always displayed and that the description, subject, keywords, etc. will NOT be displayed if you do not include them in the :CustomDatasetSummaryFields.

```
curl http://localhost:8080/api/admin/settings/:CustomDatasetSummaryFields -X PUT -d
'producer,subtitle, alternativeTitle'
```

You have to put the datasetFieldType name attribute in the :CustomDatasetSummaryFields setting for this to work.

The default fields are `dsDescription,subject,keyword,publication,notesText`.

This setting can be retrieved via API. See *Get Summary Field Names* in the API Guide.

:AllowApiTokenLookupViaApi

The Dataverse Software 4.8.1 and below allowed API Token lookup via API but for better security this has been disabled by default. Set this to true if you really want the old behavior.

```
curl -X PUT -d 'true' http://localhost:8080/api/admin/settings/:AllowApiTokenLookupViaApi
```

:ProvCollectionEnabled

Enable the collection of provenance metadata on your Dataverse installation via the provenance popup.

```
curl -X PUT -d 'true' http://localhost:8080/api/admin/settings/:ProvCollectionEnabled
```

:MetricsCacheTimeoutMinutes

Sets how long a cached metrics result is used before re-running the query for a request. This timeout is only applied to some of the metrics that query the current state of the system, previous months queries are cached indefinitely. See *Metrics API* for more info. The default timeout value is 7 days (10080 minutes).

```
curl -X PUT -d 10080 http://localhost:8080/api/admin/settings/:MetricsCacheTimeoutMinutes
```

:MDCLogPath

Sets the path where the raw Make Data Count logs are stored before being processed. If not set, no logs will be created for Make Data Count. See also the *Make Data Count* section of the Admin Guide.

```
curl -X PUT -d '/usr/local/payara7/glassfish/domains/domain1/logs' http://localhost:8080/
api/admin/settings/:MDCLogPath
```

:DisplayMDCMetrics

`:DisplayMDCMetrics` can be set to false to disable display of MDC metrics (e.g. to enable collection of MDC metrics for some period prior to completing the set-up of Counter and performing the other steps described in the *Make Data Count* section of the Admin Guide).

```
curl -X PUT -d 'false' http://localhost:8080/api/admin/settings/:DisplayMDCMetrics
```

:MDCStartDate

It is possible to display MDC metrics (as of the start date of MDC logging) along with legacy download counts, generated before MDC was enabled. This is enabled via the new setting *:MDCStartDate* that specifies the cut-over date. If a dataset has any legacy access counts collected prior to that date, those numbers will be displayed in addition to the MDC views and downloads recorded since then. (Nominally, this date should be when your installation started logging MDC metrics but it can be any date after that if desired.)

```
curl -X PUT -d '2019-10-01' http://localhost:8080/api/admin/settings/:MDCStartDate
```

:ContactFeedbackMessageSizeLimit

Maximum length of the text body that can be sent to the contacts of a Collection, Dataset, or DataFile. Setting this limit to Zero will denote unlimited length.

```
curl -X PUT -d 1080 http://localhost:8080/api/admin/settings/:ContactFeedbackMessageSizeLimit
```

:Languages

Sets which languages should be available. If there is more than one, a dropdown is displayed in the header.

See *Internationalization* for a curl example and related settings.

:MetadataLanguages

Sets which languages can be used when entering dataset metadata.

See *Internationalization* for further discussion, a curl example, and related settings.

:InheritParentRoleAssignments

`:InheritParentRoleAssignments` can be set to a comma-separated list of role aliases or '*' (all) to cause newly created Dataverse collections to inherit the set of users and/or internal groups who have assignments for those role(s) on the parent Dataverse collection, i.e. those users/groups will be assigned the same role(s) on the new Dataverse collection (in addition to the creator of the new Dataverse collection having an admin role). This can be helpful in situations where multiple organizations are sharing one Dataverse installation. The default, if `:InheritParentRoleAssignments` is not set is for the creator of the new Dataverse collection to be the only one assigned a role.

```
curl -X PUT -d 'admin, curator' http://localhost:8080/api/admin/settings/
:InheritParentRoleAssignments    or    curl -X PUT -d '*' http://localhost:8080/api/admin/
settings/:InheritParentRoleAssignments
```

:ChronologicalDateFacets

Unlike other facets, those indexed by Date/Year are sorted chronologically by default, with the most recent value first. To have them sorted by number of hits, e.g. with the year with the most results first, set this to false

If you don't want date facets to be sorted chronologically, set:

```
curl -X PUT -d 'false' http://localhost:8080/api/admin/settings/:ChronologicalDateFacets
```

:CustomZipDownloadServiceUrl

The location of the "Standalone Zipper" service. If this option is specified, the Dataverse installation will be redirecing bulk/multi-file zip download requests to that location, instead of serving them internally. See *Standalone "Zipper" Service Tool* of the Advanced Installation guide for information on how to install the external zipper. (This is still an **experimental** feature, as of Dataverse Software 5.0).

To enable redirects to the zipper installed on the same server as the main Dataverse Software application:

```
curl -X PUT -d '/cgi-bin/zipdownload' http://localhost:8080/api/admin/settings/
:CustomZipDownloadServiceUrl
```

To enable redirects to the zipper on a different server:

```
curl -X PUT -d 'https://zipper.example.edu/cgi-bin/zipdownload' http://localhost:8080/
api/admin/settings/:CustomZipDownloadServiceUrl
```

:CreateDataFilesMaxErrorsToDisplay

Number of errors to display to the user when creating DataFiles from a file upload. It defaults to 5 errors.

```
curl -X PUT -d '1' http://localhost:8080/api/admin/settings/:CreateDataFilesMaxErrorsToDisplay
```

:WorkflowsAdminIpWhitelist

A semicolon-separated list of IP addresses from which workflow resume requests are honored. By default, the Dataverse installation honors resume requests from localhost only (`127.0.0.1;::1`). This setting allows for preventing unauthorized resuming of workflows.

```
curl -X PUT -d '127.0.0.1;::1;192.168.0.1' http://localhost:8080/api/admin/settings/
:WorkflowsAdminIpWhitelist
```

See *Workflow Admin section* for more details and context.

:PrePublishDatasetWorkflowId

The identifier of the workflow to be executed prior to dataset publication. This pre-publish workflow is useful for preparing a dataset for public access (e.g., moving files, checking metadata) or starting an approval process.

```
curl -X PUT -d '1' http://localhost:8080/api/admin/settings/:PrePublishDatasetWorkflowId
```

See *Workflow Admin section* for more details and context.

:PostPublishDatasetWorkflowId

The identifier of the workflow to be executed after a dataset has been successfully published. This post-publish workflow is useful for actions such as sending notifications about the newly published dataset or archiving.

```
curl -X PUT -d '2' http://localhost:8080/api/admin/settings/:PostPublishDatasetWorkflowId
```

See *Workflow Admin section* for more details and context.

:PublishDatasetDisclaimerText

The text displayed to the user that must be acknowledged prior to publishing a Dataset. When not set the acknowledgment is not required nor displayed.

```
curl -X PUT -d "By publishing this dataset, I fully accept all legal responsibility
for ensuring that the deposited content is:  anonymized, free of copyright violations,
and contains data that is computationally reusable. I understand and agree that any
violation of these conditions may result in the immediate removal of the dataset
by the repository without prior notice." http://localhost:8080/api/admin/settings/
:PublishDatasetDisclaimerText
```

There is a similar setting called *:DatasetPublishPopupCustomText* that also makes text appear on the popup when publishing, but it is only informational. There is no checkbox to click.

See also *Show Disclaimer for Publishing Datasets* in the API Guide.

:SubmitForReviewDatasetDisclaimerText

The text displayed to the user that must be acknowledged prior to submitting a Dataset for review. When not set the acknowledgment is not required nor displayed.

```
curl -sS -X PUT -d 'I agree to the following:<br/>1. My submission has been fully
anonymized (required for all human subject'\''s datasets).<br/>2. My submission does
not violate any known copyright laws.<br/>3. I understand that I am liable for any and
all violations of the Harvard Repository <a href=https://support.dataverse.harvard.edu/
harvard-dataverse-general-terms-use>Terms of Use.</a>' http://localhost:8080/api/admin/
settings/:SubmitForReviewDatasetDisclaimerText
```

See also *Show Disclaimer for Submitting For Review Datasets* in the API Guide.

:BagItHandlerEnabled

Part of the database settings to configure the BagIt file handler. Enables the BagIt file handler. By default, the handler is disabled.

```
curl -X PUT -d 'true' http://localhost:8080/api/admin/settings/:BagItHandlerEnabled
```

:BagValidatorJobPoolSize

Part of the database settings to configure the BagIt file handler. The number of threads the checksum validation class uses to validate a single zip file. Defaults to 4 threads

```
curl -X PUT -d '10' http://localhost:8080/api/admin/settings/:BagValidatorJobPoolSize
```

:BagValidatorMaxErrors

Part of the database settings to configure the BagIt file handler. The maximum number of errors allowed before the validation job aborts execution. This is to avoid processing the whole BagIt package. Defaults to 5 errors.

```
curl -X PUT -d '2' http://localhost:8080/api/admin/settings/:BagValidatorMaxErrors
```

:BagValidatorJobWaitInterval

Part of the database settings to configure the BagIt file handler. This is the period in seconds to check for the number of errors during validation. Defaults to 10.

```
curl -X PUT -d '60' http://localhost:8080/api/admin/settings/:BagValidatorJobWaitInterval
```

:ArchiverClassName

Your Dataverse installation can export archival "Bag" files to an extensible set of storage systems (see *BagIt Export* above for details about this and for further explanation of the other archiving related settings below). This setting specifies which storage system to use by identifying the particular Java class that should be run. Current configuration options include DuraCloudSubmitToArchiveCommand, LocalSubmitToArchiveCommand, GoogleCloudSubmitToArchiveCommand, and S3SubmitToArchiveCommand.

For examples, see the specific configuration above in *BagIt Export*.

:ArchiveOnlyIfEarlierVersionsAreArchived

This setting, if true, only allows creation of an archival Bag for a dataset version if all prior versions have been successfully archived. The default is false (any version can be archived independently as long as other settings allow it)

:ArchiverSettings

Each Archiver class may have its own custom settings. Along with setting which Archiver class to use, one must use this setting to identify which setting values should be sent to it when it is invoked. The value should be a comma-separated list of setting names. For example, the LocalSubmitToArchiveCommand only uses the :BagItLocalPath setting. To allow the class to use that setting, this setting must set as:

```
curl -X PUT -d ':BagItLocalPath' http://localhost:8080/api/admin/settings/
:ArchiverSettings
```

:BagGeneratorThreads

An archiver setting shared by several implementations (e.g. DuraCloud, Google, and Local) that can make Bag generation use fewer or more threads in zipping datafiles that the default of 2

```
curl http://localhost:8080/api/admin/settings/:BagGeneratorThreads -X PUT -d '8'
```

:DuraCloudHost

:DuraCloudPort

:DuraCloudContext

These three settings define the host, port, and context used by the DuraCloudSubmitToArchiveCommand. :DuraCloudHost is required. The other settings have default values as noted in the *Duracloud Configuration* section above.

:BagItLocalPath

This is the local file system path to be used with the LocalSubmitToArchiveCommand class. It is recommended to use an absolute path. See the *Local Path Configuration* section above.

:GoogleCloudBucket

:GoogleCloudProject

These are the bucket and project names to be used with the GoogleCloudSubmitToArchiveCommand class. Further information is in the *Google Cloud Configuration* section above.

:S3ArchiverConfig

This is the JSON configuration object setting to be used with the S3SubmitToArchiveCommand class. Further information is in the *S3 Configuration* section above.

:InstallationName

As explained under *Branding Your Installation*, by default, the name of the root Dataverse collection is used as the "brand name" of the installation, i.e. in emails and metadata exports. If set, `:InstallationName` overrides this default, allowing the root collection name and brandname to be set independently. (Note that, since metadata export files are cached, they will have to be reexported (see *Metadata Export*) before they incorporate a change in this setting.)

:ExportInstallationAsDistributorOnlyWhenNotSet

In the DDI metadata exports, the default behavior is to always add the repository (using its brandname - the root collection name or the value of *:InstallationName*) to the stdyDscr/distStmt/distrbtr element. If this setting is true, this will only be done when a Distributor is not already defined in the Dataset metadata. (Note that, since metadata export files are cached, they will have to be reexported (see *Metadata Export*) before they incorporate a change in this setting.)

:AnonymizedFieldTypeNames

A comma-separated list of field type names that should be 'withheld' when dataset access occurs via a Private Url with Anonymized Access (e.g. to support anonymized review). A suggested minimum includes author, datasetContact, and contributor, but additional fields such as depositor, grantNumber, and publication might also need to be included.

```
curl -X PUT -d 'author, datasetContact, contributor, depositor, grantNumber, publication'
http://localhost:8080/api/admin/settings/:AnonymizedFieldTypeNames
```

:DatasetChecksumValidationSizeLimit

Setting `DatasetChecksumValidationSizeLimit` to a threshold in bytes, disables the checksum validation while publishing for any dataset size greater than the limit.

For example, if you want your Dataverse installation to skip validation for any dataset larger than 5 GB while publishing, use this setting:

```
curl -X PUT -d 5000000000 http://localhost:8080/api/admin/settings/
:DatasetChecksumValidationSizeLimit
```

When this option is used to disable the checksum validation, it's strongly recommended to perform periodic asynchronous checks via the integrity API

Refer to "Physical Files Validation in a Dataset" API *Physical Files Validation in a Dataset* section of our *Native API* documentation.

Also refer to the "Datafile Integrity" API *Datafile Integrity*

:DataFileChecksumValidationSizeLimit

Setting `DataFileChecksumValidationSizeLimit` to a threshold in bytes, disables the checksum validation while publishing for any datafiles greater than the limit.

For example, if you want your Dataverse installation to skip validation for any data files larger than 2 GB while publishing, use this setting:

```
curl -X PUT -d 2000000000 http://localhost:8080/api/admin/settings/
:DataFileChecksumValidationSizeLimit
```

When this option is used to disable the checksum validation, it's strongly recommended to perform periodic asynchronous checks via the integrity API

Refer to "Physical Files Validation in a Dataset" API *Physical Files Validation in a Dataset* section of our *Native API* documentation.

Also refer to the "Datafile Integrity" API *Datafile Integrity*

:SendNotificationOnDatasetCreation

A boolean setting that, if true, will send an email and notification to users when a Dataset is created. Messages go to those, other than the dataset creator, who have the ability/permission necessary to publish the dataset. The intent of this functionality is to simplify tracking activity and planning to follow-up contact.

```
curl -X PUT -d true http://localhost:8080/api/admin/settings/:SendNotificationOnDatasetCreation
```

:CVocConf

The `:CVocConf` database setting is used to allow metadatablock fields to look up values in external vocabulary services. For example, you could configure the "Author Affiliation" field to look up organizations in the Research Organization Registry (ROR). For a high-level description of this feature, see *Using External Vocabulary Services* in the Admin Guide.

The expected format for the `:CVocConf` database setting is JSON but the details are not documented here. Instead, please refer to docs/readme.md in the https://github.com/gdcc/dataverse-external-vocab-support repo.

That repository also includes scripts that implement the lookup for specific service protocols, a JSON Schema for validating the structure required by this setting, and an example metadatablock with a sample `:CVocConf` config that associates fields in the example block with ORCID and SKOSMOS based services.

The commands below should give you an idea of how to load the configuration, but you'll want to study the examples and make decisions about which configuration to use:

```
wget https://gdcc.github.io/dataverse-external-vocab-support/examples/config/cvoc-conf.
json
```

```
curl -X PUT --upload-file cvoc-conf.json http://localhost:8080/api/admin/settings/
:CVocConf
```

Since external vocabulary scripts can change how fields are indexed (storing an identifier and name and/or values in different languages), updating the Solr schema as described in *Updating the Solr Schema* should be done after adding new scripts to your configuration.

:ControlledVocabularyCustomJavaScript

`:ControlledVocabularyCustomJavaScript` allows a JavaScript file to be loaded into the dataset page for the purpose of showing controlled vocabulary as a list (with optionally translated values) such as author names.

To specify the URL for a custom script `covoc.js` to be loaded from an external site:

```
curl -X PUT -d 'https://example.com/js/covoc.js' http://localhost:8080/api/admin/
settings/:ControlledVocabularyCustomJavaScript
```

To remove the custom script URL:

```
curl -X DELETE http://localhost:8080/api/admin/settings/:ControlledVocabularyCustomJavaScript
```

Please note that *:CVocConf* is a better option if the list is large or needs to be searchable from an external service using protocols such as SKOSMOS.

:AllowedCurationLabels

A JSON Object containing lists of allowed labels (up to 32 characters, spaces allowed) that can be set, via API or UI by users with the permission to publish a dataset. The set of labels allowed for datasets can be selected by a superuser - via the Dataverse collection page (Edit/General Info) or set via API call. The labels in a set should correspond to the states in an organization's curation process and are intended to help users/curators track the progress of a dataset through a defined curation process. A dataset may only have one label at a time and if a label is set, it will be removed at publication time. This functionality is disabled when this setting is empty/not set. Each set of labels is identified by a curationLabelSet name and a JSON Array of the labels allowed in that set.

```
curl -X PUT -d '{"Standard Process":["Author contacted", "Privacy Review", "Awaiting
paper publication", "Final Approval"], "Alternate Process":["State 1","State 2","State
3"]}' http://localhost:8080/api/admin/settings/:AllowedCurationLabels
```

If the Dataverse Installation supports multiple languages, the curation label translations should be added to the `CurationLabels` properties files. (See *Internationalization* for more on properties files and internationalization in general.) Since the Curation labels are free text, while creating the key, it has to be converted to lowercase, replace space with underscore.

Example:

```
standard_process=Standard Process
author_contacted=Author contacted
```

:AllowCustomTermsOfUse

By default, custom terms of data use and access can be specified after selecting "Custom Terms" from the License/DUA dropdown on the Terms tab. When `:AllowCustomTermsOfUse` is set to `false` the "Custom Terms" item is not made available to the depositor.

```
curl -X PUT -d false http://localhost:8080/api/admin/settings/:AllowCustomTermsOfUse
```

:MaxEmbargoDurationInMonths

This setting controls whether embargoes are allowed in a Dataverse instance and can limit the maximum duration users are allowed to specify. A value of 0 months or non-existent setting indicates embargoes are not supported. A value of -1 allows embargoes of any length. Any other value indicates the maximum number of months (from the current date) a user can enter for an embargo end date. This limit will be enforced in the popup dialog in which users enter the embargo date. For example, to set a two year maximum:

```
curl -X PUT -d 24 http://localhost:8080/api/admin/settings/:MaxEmbargoDurationInMonths
```

:MinRetentionDurationInMonths

This setting controls whether retention periods are allowed in a Dataverse instance and can limit the minimum duration users are allowed to specify. A value of 0 months or non-existent setting indicates retention periods are not supported. A value of -1 allows retention periods of any length. Any other value indicates the minimum number of months (from the current date) a user can enter for a retention period end date. This limit will be enforced in the popup dialog in which users enter the retention period end date. For example, to set a ten year minimum:

```
curl -X PUT -d 120 http://localhost:8080/api/admin/settings/:MinRetentionDurationInMonths
```

:DataverseMetadataValidatorScript

An optional external script that validates Dataverse collection metadata as it's being updated or published. The script provided should be an executable that takes a single command line argument, the name of the file containing the metadata exported in the native json format. I.e., Dataverse application will be exporting the collection metadata in json format, saving it in a temp file, and passing the name of the temp file to the validation script as the command line argument. The script should exit with a non-zero error code if the validation fails. If that happens, a failure message (customizable in the next two settings below, *:DataverseMetadataPublishValidationFailureMsg* and *:DataverseMetadataUpdateValidationFailureMsg*) will be shown to the user.

For example, once the following setting is created:

```
curl -X PUT -d /usr/local/bin/dv_validator.sh http://localhost:8080/api/admin/settings/
:DataverseMetadataValidatorScript
```

:DataverseMetadataPublishValidationFailureMsg

Specifies a custom error message shown to the user when a Dataverse collection fails an external metadata validation (as specified in the setting above) during an attempt to publish. If not specified, the default message "This dataverse collection cannot be published because it has failed an external metadata validation test" will be used.

For example:

```
curl -X PUT -d "This content needs to go through an additional review by the
Curation Team before it can be published." http://localhost:8080/api/admin/settings/
:DataverseMetadataPublishValidationFailureMsg
```

:DataverseMetadataUpdateValidationFailureMsg

Same as above, but specifies a custom error message shown to the user when an external metadata validation check fails during an attempt to modify a Dataverse collection. If not specified, the default message "This dataverse collection cannot be updated because it has failed an external metadata validation test" will be used.

:DatasetMetadataValidatorScript

An optional external script that validates dataset metadata during publishing. The script provided should be an executable that takes a single command line argument, the name of the file containing the metadata exported in the native json format. I.e., Dataverse application will be exporting the dataset metadata in json format, saving it in a temp file, and passing the name of the file to the validation script as the command line argument. The script should exit with a non-zero error code if the validation fails. If that happens, the dataset is left unpublished, and a failure message (customizable in the next setting below, *:DatasetMetadataValidationFailureMsg*) will be shown to the user.

For example:

```
curl -X PUT -d /usr/local/bin/ds_validator.sh http://localhost:8080/api/admin/settings/
:DatasetMetadataValidatorScript
```

In some ways this duplicates a workflow mechanism, since it is possible to define a workflow with additional validation steps. But please note that the important difference is that this external validation happens *synchronously*, while the user is wating; while a workflow is performed asynchronously with a lock placed on the dataset. This can be useful to some installations, in some situations. But it also means that the script provided should be expected to always work reasonably fast - ideally, in seconds, rather than minutes, etc.

:DatasetMetadataValidationFailureMsg

Specifies a custom error message shown to the user when a dataset fails an external metadata validation (as specified in the setting above) during an attempt to publish. If not specified, the default message "This dataset cannot be published because it has failed an external metadata validation test" will be used.

For example:

```
curl -X PUT -d "This content needs to go through an additional review by the
Curation Team before it can be published." http://localhost:8080/api/admin/settings/
:DatasetMetadataValidationFailureMsg
```

:ExternalValidationAdminOverride

When set to `true`, this setting allows a superuser to publish and/or update Dataverse collections and datasets bypassing the external validation checks (specified by the settings above). In an event where an external script is reporting validation failures that appear to be in error, this option gives an admin with superuser privileges a quick way to publish the dataset or update a collection for the user.

:FileCategories

Overrides the default list of file categories that is used in the UI when adding tags to files. The default list is Documentation, Data, and Code.

This setting is a comma-separated list of the new tags.

To override the default list with Docs, Data, Code, and Workflow:

```
curl -X PUT -d 'Docs,Data,Code,Workflow' http://localhost:8080/api/admin/settings/
:FileCategories
```

To remove the override and go back to the default list:

```
curl -X PUT -d '' http://localhost:8080/api/admin/settings/:FileCategories
```

:ShowMuteOptions

Allows users to mute notifications by showing additional configuration options in the Notifications tab of the account page (see *Notifications* in the User Guide). By default, this setting is "false" and users cannot mute any notifications (this feature is not shown in the user interface).

For configuration details, see *Letting Users Manage Notifications*.

:AlwaysMuted

Overrides the default empty list of always muted notifications. Always muted notifications cannot be unmuted by the users. Always muted notifications are not shown in the notification settings for the users.

For configuration details, see *Letting Users Manage Notifications*.

:NeverMuted

Overrides the default empty list of never muted notifications. Never muted notifications cannot be muted by the users. Always muted notifications are grayed out and are not adjustable by the user.

For configuration details, see *Letting Users Manage Notifications*.

:LDNMessageHosts

The comma-separated list of hosts allowed to send Dataverse Linked Data Notification messages. See *Linked Data Notification API* for details. * allows messages from anywhere (not recommended for production). By default, messages are not accepted from anywhere.

:LDN_TARGET

The URL of an LDN Inbox to which the LDN Announce workflow step will send messages. See *Workflows* for details.

:LDNAnnounceRequiredFields

The list of parent dataset field names for which the LDN Announce workflow step should send messages. See *Workflows* for details.

:GlobusAppUrl

The URL where the dataverse-globus "transfer" app has been deployed to support Globus integration. See *Globus File Transfer* for details.

:GlobusPollingInterval

The interval in seconds between Dataverse calls to Globus to check on upload progress. Defaults to 50 seconds (or to 10 minutes, when the `globus-use-experimental-async-framework` feature flag is enabled). See *Globus File Transfer* for details.

:GlobusBatchLookupSize

In the initial implementation, when files were added to the dataset upon completion of a Globus upload task, Dataverse would make a separate Globus API call to look up the size of every new file. This proved to be a significant bottleneck at Harvard Dataverse with users transferring batches of many thousands of files (this in turn was made possible by the Globus improvements in v6.4). An optimized lookup mechanism was added in response, where the Globus Service makes a listing API call on the entire remote folder, then populates the file sizes for all the new file entries before passing them to the Ingest service. This approach however may in fact slow things down in a scenario where there are already thousands of files in the Globus folder for the dataset, and only a small number of new files are being added. To address this, the number of files in a batch for which this method should be used was made configurable. If not set, it will default to 50 (a completely arbitrary number). Setting it to 0 will always use this method with Globus uploads. Setting it to some very large number will disable it completely. This was made a database setting, as opposed to a JVM option, in order to make it configurable in real time.

:GlobusSingleFileTransfer

A true/false option to add a Globus transfer option to the file download menu which is not yet fully supported in the dataverse-globus app. See *Globus File Transfer* for details.

:WebloaderUrl

The URL of *dvuploader <https://github.com/gdcc/dvwebloader>*'s HTML file when dvuploader is enabled in *:UploadMethods*.

To use the current GDCC version directly:

```
curl -X PUT -d 'https://gdcc.github.io/dvwebloader/src/dvwebloader.html' http://
localhost:8080/api/admin/settings/:WebloaderUrl
```

:CategoryOrder

A comma separated list of Category/Tag names defining the order in which files with those tags should be displayed. The setting can include custom tag names along with the pre-defined tags (Documentation, Data, and Code are the defaults but the *:FileCategories* setting can be used to use a different set of tags). The default is category ordering disabled.

```
curl -X PUT -d 'Documentation,Data,Code' http://localhost:8080/api/admin/settings/
:CategoryOrder
```

:OrderByFolder

A true(default)/false option determining whether datafiles listed on the dataset page should be grouped by folder.

```
curl -X PUT -d true http://localhost:8080/api/admin/settings/:OrderByFolder
```

:AllowUserManagementOfOrder

A true/false (default) option determining whether the dataset datafile table display includes checkboxes enabling users to turn folder ordering and/or category ordering (if an order is defined by :CategoryOrder) on and off dynamically.

```
curl -X PUT -d true http://localhost:8080/api/admin/settings/:AllowUserManagementOfOrder
```

:UseStorageQuotas

Enables storage use quotas in collections. See the *Native API* for details.

:StoreIngestedTabularFilesWithVarHeaders

With this setting enabled, tabular files produced during Ingest will be stored with the list of variable names added as the first tab-delimited line. As the most significant effect of this feature, Access API will be able to take advantage of Direct Download for tab. files saved with these headers on S3 - since they no longer have to be generated and added to the streamed file on the fly.

The setting is `false` by default, preserving the legacy behavior.

:RateLimitingDefaultCapacityTiers

Number of calls allowed per hour if the specific command is not configured. The values represent the number of calls per hour per user for tiers 0,1,… A value of -1 can be used to signify no rate limit. Also, by default, a tier not defined would receive a default of no limit.

See also *Rate Limiting*.

:RateLimitingCapacityByTierAndAction

JSON object specifying the rate by tier and a list of actions (commands). This allows for more control over the rate limit of individual API command calls. In the following example, calls made by a guest user (tier 0) for API GetLatestPublishedDatasetVersionCommand is further limited to only 10 calls per hour, while an authenticated user (tier 1) will be able to make 30 calls per hour to the same API.

```
{"rateLimits":[
  {"tier": 0, "limitPerHour": 10, "actions": ["GetLatestPublishedDatasetVersionCommand",
↪"GetPrivateUrlCommand", "GetDatasetCommand", "GetLatestAccessibleDatasetVersionCommand
↪"]},
  {"tier": 0, "limitPerHour": 1, "actions": ["CreateGuestbookResponseCommand",
↪"UpdateDatasetVersionCommand", "DestroyDatasetCommand", "DeleteDataFileCommand",
↪"FinalizeDatasetPublicationCommand", "PublishDatasetCommand"]},
  {"tier": 1, "limitPerHour": 30, "actions": ["CreateGuestbookResponseCommand",
↪"GetLatestPublishedDatasetVersionCommand", "GetPrivateUrlCommand", "GetDatasetCommand",
↪ "GetLatestAccessibleDatasetVersionCommand", "UpdateDatasetVersionCommand",
↪"DestroyDatasetCommand", "DeleteDataFileCommand", "FinalizeDatasetPublicationCommand",
↪"PublishDatasetCommand"]}
]}
```

See also *Rate Limiting*.

:GetExternalSearchUrl

:GetExternalSearchName

:PostExternalSearchUrl

:PostExternalSearchName

Experimental - settings for Search Services. See *Search Services*.

6.6 Big Data Support

Big data support includes some experimental options.

Contents:

- *S3 Direct Upload and Download*
  - *Features that are Disabled if S3 Direct Upload is Enabled*

    - *Allow CORS for S3 Buckets*
        * *Get Current CORS Policy on Bucket*
        * *Set CORS Policy on Bucket*
    - *S3 Tags and Direct Upload*
- *Trusted Remote Storage with the* `remote` *Store Type*
- *Globus File Transfer*

Various components will need to be installed and/or configured for big data support via the methods described below.

6.6.1 S3 Direct Upload and Download

A lightweight option for supporting file sizes beyond a few gigabytes - a size that can cause performance issues when uploaded through a Dataverse installation itself - is to configure an S3 store to provide direct upload and download via 'pre-signed URLs'. When these options are configured, file uploads and downloads are made directly to and from a configured S3 store using secure (https) connections that enforce a Dataverse installation's access controls. (The upload and download URLs are signed with a unique key that only allows access for a short time period and a Dataverse installation will only generate such a URL if the user has permission to upload/download the specific file in question.)

This option can handle files >300GB and could be appropriate for files up to a TB or larger. Other options can scale farther, but this option has the advantages that it is simple to configure and does not require any user training - uploads and downloads are done via the same interface as normal uploads to a Dataverse installation.

To configure these options, an administrator must set two JVM options for the Dataverse installation using the same process as for other configuration options:

```
./asadmin create-jvm-options "-Ddataverse.files.<id>.download-redirect=true"
```

```
./asadmin create-jvm-options "-Ddataverse.files.<id>.upload-redirect=true"
```

With multiple stores configured, it is possible to configure one S3 store with direct upload and/or download to support large files (in general or for specific Dataverse collections) while configuring only direct download, or no direct access for another store.

The direct upload option now switches between uploading the file in one piece (up to 1 GB by default) and sending it as multiple parts. The default can be changed by setting:

```
./asadmin create-jvm-options "-Ddataverse.files.<id>.min-part-size=<size in bytes>"
```

For AWS, the minimum allowed part size is 5*1024*1024 bytes and the maximum is 5 GB (5*1024**3). Other providers may set different limits.

It is also possible to set file upload size limits per store. See the :MaxFileUploadSizeInBytes setting described in the *Configuration* guide.

At present, one potential drawback for direct-upload is that files are only partially 'ingested' - tabular and FITS files are processed, but zip files are not unzipped, and the file contents are not inspected to evaluate their mimetype. This could be appropriate for large files, or it may be useful to completely turn off ingest processing for performance reasons (ingest processing requires a copy of the file to be retrieved by the Dataverse installation from the S3 store). A store using direct upload can be configured to disable all ingest processing for files above a given size limit:

```
./asadmin create-jvm-options "-Ddataverse.files.<id>.ingestsizelimit=<size in bytes>"
```

Features that are Disabled if S3 Direct Upload is Enabled

The following features are disabled when S3 direct upload is enabled.

- Unzipping of zip files. (See *Compressed Files*.)
- Detection of file type based on JHOVE and custom code that reads the first few bytes except for the refinement of Stata file types to include the version. (See *Redetect File Type*.)
- Extraction of metadata from FITS files. (See *Astronomy (FITS)*.)
- Creation of NcML auxiliary files (See *NetCDF and HDF5*.)
- Extraction of a geospatial bounding box from NetCDF and HDF5 files (see *NetCDF and HDF5*) unless *dataverse.netcdf.geo-extract-s3-direct-upload* is set to true.

Allow CORS for S3 Buckets

IMPORTANT: This additional step of allowing cross-site request to your S3 buckets is required to enable direct uploads via a Dataverse installation, direct download to work with *file previewers*, or direct upload to work with *DVWebloader*.

To successfully enable direct uploads (e.g. *Folder Upload*) or direct downloads (e. g. consumed by previewers), you must both:

- Enable CORS in Dataverse (see *Cross-Origin Resource Sharing (CORS)*).
- Configure a matching/compatible CORS policy on each S3 bucket (and any CDN/proxy in front of it) that will be used.

NOTE: Make sure the bucket's CORS configuration `AllowedOrigins` is at least as permissive as the origins you configure in *dataverse.cors.origin*. If the bucket allows the wildcard `*` but the Dataverse application only allows a subset, the browser will still enforce the more restrictive application response!

Detailed information for the most common S3 admin tools around CORS:

- AWS
- s3cmd

Get Current CORS Policy on Bucket

If you'd like to check the CORS configuration on your bucket before making changes:

AWS CLI

```
aws s3api get-bucket-cors --bucket <BUCKET_NAME>
```

Set CORS Policy on Bucket

The examples below shows how to enable CORS rules (to support upload and download) on a bucket.

Note: You may want to limit the `AllowedOrigins` and/or `AllowedHeaders` further. GDCC/dataverse-previewers has some additional information about doing this.

Both JSON and XML format are explained in detail in AWS Docs.

AWS CLI

Create a file `cors.json` as follows:

```
{
    "CORSRules": [
        {
            "AllowedOrigins": ["*"],
            "AllowedHeaders": ["*"],
            "AllowedMethods": ["PUT", "GET"],
            "ExposeHeaders": ["ETag", "Accept-Ranges", "Content-Encoding", "Content-Range
↪"]
        }
    ]
}
```

Proceed with making the changes:

```
aws s3api put-bucket-cors --bucket <BUCKET_NAME> --cors-configuration file://cors.json
```

Alternatively, you can enable CORS using the AWS S3 web interface, using json-encoded rules as in the example above.

```
<?xml version="1.0" encoding="UTF-8"?>
<CORSConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <CORSRule>
        <AllowedOrigin>*</AllowedOrigin>
        <AllowedHeader>*</AllowedHeader>
        <AllowedMethod>PUT</AllowedMethod>
        <AllowedMethod>GET</AllowedMethod>
        <ExposeHeader>ETag</ExposeHeader>
        <ExposeHeader>Accept-Ranges</ExposeHeader>
        <ExposeHeader>Content-Encoding</ExposeHeader>
        <ExposeHeader>Content-Range</ExposeHeader>
    </CORSRule>
</CORSConfiguration>
```

Proceed with making the changes:

```
mc cors set <STORE_NAME>/<BUCKET_NAME> ./cors.xml
```

S3 Tags and Direct Upload

Since the direct upload mechanism creates the final file rather than an intermediate temporary file, user actions, such as neither saving or canceling an upload session before closing the browser page, can leave an abandoned file in the store. The direct upload mechanism attempts to use S3 tags to aid in identifying/removing such files. Upon upload, files are given a "dv-state":"temp" tag which is removed when the dataset changes are saved and new files are added in the Dataverse installation. Note that not all S3 implementations support tags. With such stores, direct upload may not work and you might need to disable tagging. For details, see *S3 Tagging* in the Installation Guide.

6.6.2 Trusted Remote Storage with the `remote` Store Type

For very large, and/or very sensitive data, it may not make sense to transfer or copy files to Dataverse at all. The experimental `remote` store type in the Dataverse software now supports this use case.

With this storage option Dataverse stores a URL reference for the file rather than transferring the file bytes to a store managed directly by Dataverse. Basic configuration for a remote store is described at *File Storage* in the Configuration Guide.

Once the store is configured, it can be assigned to a collection or individual datasets as with other stores. In a dataset using this store, users can reference remote files which will then appear the same basic way as other datafiles.

Currently, remote files can only be added via the API. Users can also upload smaller files via the UI or API which will be stored in the configured base store.

If the store has been configured with a remote-store-name or remote-store-url, the dataset file table will include this information for remote files. These provide a visual indicator that the files are not managed directly by Dataverse and are stored/managed by a remote trusted store.

Rather than sending the file bytes, metadata for the remote file is added using the "jsonData" parameter. jsonData normally includes information such as a file description, tags, provenance, whether the file is restricted, etc. For remote references, the jsonData object must also include values for:

- "storageIdentifier" - String, as specified in prior calls
- "fileName" - String
- "mimeType" - String
- fixity/checksum: either:
  - "md5Hash" - String with MD5 hash value, or
  - "checksum" - Json Object with "@type" field specifying the algorithm used and "@value" field with the value from that algorithm, both Strings

The allowed checksum algorithms are defined by the edu.harvard.iq.dataverse.DataFile.CheckSumType class and currently include MD5, SHA-1, SHA-256, and SHA-512

(The remote store leverages the same JSON upload syntax as the last step in direct upload to S3 described in the *Adding the Uploaded file to the Dataset* section of the *Direct DataFile Upload/Replace API*.)

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK27U7YBV
export JSON_DATA='{"description":"My description.","directoryLabel":"data/subdir1",
↪"categories":["Data"], "restrict":"false", "storageIdentifier":"trs://images/dataverse_
↪project_logo.svg", "fileName":"dataverse_logo.svg", "mimeType":"image/svg+xml",
↪"checksum": {"@type": "SHA-1", "@value": "123456"}}'

curl -X POST -H "X-Dataverse-key: $API_TOKEN" "$SERVER_URL/api/datasets/:persistentId/
↪add?persistentId=$PERSISTENT_IDENTIFIER" -F "jsonData=$JSON_DATA"
```

The variant allowing multiple files to be added once that is discussed in the *Direct DataFile Upload/Replace API* document can also be used.

Considerations:

- Remote stores are configured with a base-url which limits what files can be referenced, i.e. the absolute URL for the file is <base-url>/<path in storageidentifier>.

- The current store will not prevent you from providing a relative URL that results in a 404 when resolved. (I.e. if you make a typo). You should check to make sure the file exists at the location you specify - by trying to download in Dataverse, by checking to see that Dataverse was able to get the file size (which it does by doing a HEAD call to that location), or just manually trying the URL in your browser.
- Admins are trusting the organization managing the site/service at base-url to maintain the referenced files for as long as the Dataverse instance needs them. Formal agreements are recommended for production
- For large files, direct-download should always be used with a remote store. (Otherwise the Dataverse will be involved in the download.)
- For simple websites, a remote store should be marked public which will turn off restriction and embargo functionality in Dataverse (since Dataverse cannot restrict access to the file on the remote website)
- Remote stores can be configured with a secret-key. This key will be used to sign URLs when Dataverse retrieves the file content or redirects a user for download. If remote service is able to validate the signature and reject invalid requests, the remote store mechanism can be used to manage restricted and embargoes files, access requests in Dataverse, etc. Dataverse contains Java code that validates these signatures which could be used, for example, to create a validation proxy in front of a web server to allow Dataverse to manage access. The secret-key is a shared secret between Dataverse and the remote service and is not shared with/is not accessible by users or those with access to user's machines.
- Sophisticated remote services may wish to register file URLs that do not directly reference the file contents (bytes) but instead direct the user to a website where further information about the remote service's download process can be found.
- Due to the current design, ingest cannot be done on remote files and administrators should disable ingest when using a remote store. This can be done by setting the ingest size limit for the store to 0 and/or using the recently added option to not perform tabular ingest on upload.
- Dataverse will normally try to access the file contents itself, i.e. for ingest (in future versions), full-text indexing, thumbnail creation, etc. This processing may not be desirable for large/sensitive data, and, for the case where the URL does not reference the file itself, would not be possible. At present, administrators should configure the relevant size limits to avoid such actions.
- The current implementation of remote stores is experimental in the sense that future work to enhance it is planned. This work may result in changes to how the store works and lead to additional work when upgrading for sites that start using this mechanism now.

To configure the options mentioned above, an administrator must set two JVM options for the Dataverse installation using the same process as for other configuration options:

```
./asadmin create-jvm-options "-Ddataverse.files.<id>.download-redirect=true"    ./asadmin
create-jvm-options "-Ddataverse.files.<id>.secret-key=somelongrandomalphanumerickeythelongerthebetter12
./asadmin create-jvm-options "-Ddataverse.files.<id>.public=true"               ./asadmin
create-jvm-options "-Ddataverse.files.<id>.ingestsizelimit=<size in bytes>"
```

6.6.3 Globus File Transfer

Note: Globus file transfer is still experimental but feedback is welcome! See *Getting Help*.

Users can transfer files via Globus into and out of datasets, or reference files on a remote Globus endpoint, when their Dataverse installation is configured to use a Globus accessible store(s) and a community-developed dataverse-globus app has been properly installed and configured.

Globus endpoints can be in a variety of places, from data centers to personal computers. This means that from within the Dataverse software, a Globus transfer can feel like an upload or a download (with Globus Personal Connect running on your laptop, for example) or it can feel like a true transfer from one server to another (from a cluster in a data center into a Dataverse dataset or vice versa).

Globus transfer uses an efficient transfer mechanism and has additional features that make it suitable for large files and large numbers of files:

- robust file transfer capable of restarting after network or endpoint failures
- third-party transfer, which enables a user accessing a Dataverse installation in their desktop browser to initiate transfer of their files from a remote endpoint (i.e. on a local high-performance computing cluster), directly to an S3 store managed by the Dataverse installation

Note: Due to differences in the access control models of a Dataverse installation and Globus and the current Globus store model, Dataverse cannot enforce per-file-access restrictions. It is therefore recommended that a store be configured as public, which disables the ability to restrict and embargo files in that store, when Globus access is allowed.

Dataverse supports three options for using Globus, two involving transfer to Dataverse-managed endpoints and one allowing Dataverse to reference files on remote endpoints. Dataverse-managed endpoints must be Globus 'guest collections' hosted on either a file-system-based endpoint or an S3-based endpoint (the latter requires use of the Globus S3 connector which requires a paid Globus subscription at the host institution). In either case, Dataverse is configured with the Globus credentials of a user account that can manage the endpoint. Users will need a Globus account, which can be obtained via their institution or directly from Globus (at no cost).

With the file-system endpoint, Dataverse does not currently have access to the file contents. Thus, functionality related to ingest, previews, fixity hash validation, etc. are not available. (Using the S3-based endpoint, Dataverse has access via S3 and all functionality normally associated with direct uploads to S3 is available.)

For the reference use case, Dataverse must be configured with a list of allowed endpoint/base paths from which files may be referenced. In this case, since Dataverse is not accessing the remote endpoint itself, it does not need Globus credentials. Users will need a Globus account in this case, and the remote endpoint must be configured to allow them access (i.e. be publicly readable, or potentially involving some out-of-band mechanism to request access (that could be described in the dataset's Terms of Use and Access).

All of Dataverse's Globus capabilities are now store-based (see the store documentation) and therefore different collections/datasets can be configured to use different Globus-capable stores (or normal file, S3 stores, etc.)

More details of the setup required to enable Globus is described in the Community Dataverse-Globus Setup and Configuration document and the references therein.

As described in that document, Globus transfers can be initiated by choosing the Globus option in the dataset upload panel. (Globus, which does asynchronous transfers, is not available during dataset creation.) Analogously, "Globus Transfer" is one of the download options in the "Access Dataset" menu and optionally the file landing page download menu (if/when supported in the dataverse-globus app).

An overview of the control and data transfer interactions between components was presented at the 2022 Dataverse Community Meeting and can be viewed in the Integrations and Tools Session Video around the 1 hr 28 min mark.

See also *Globus settings* and *Globus Stores: Globus Transfer Between Large File/Tape Archives*.

An alternative, experimental implementation of Globus polling of ongoing upload transfers has been added in v6.4. This framework does not rely on the instance staying up continuously for the duration of the transfer and saves the state information about Globus upload requests in the database. Due to its experimental nature it is not enabled by default. See the *dataverse.feature.globus-use-experimental-async-framework* feature flag and the JVM option *dataverse.files.globus-monitoring-server*.

6.7 Upgrading

When upgrading within Dataverse Software 4.x, you will need to follow the upgrade instructions for each intermediate 4.x. Similarly, when upgrading within Dataverse Software 5.x, you will need to follow the upgrade instructions for each intermediate 5.x version.

Upgrades always involve deploying the latest war file but may also include updating the schema used by Solr or other manual steps. Running database migration scripts was once required but this has been automated (see the `flyway_schema_history` database table to see migrations that have been run).

Please consult the release notes associated with each release at https://github.com/IQSS/dataverse/releases for more information.

Upgrading from DVN 3.x is actually a migration due to the many changes. Migration scripts have been checked into the source tree but as of this writing it is expected that people will require assistance running them. Please reach out per the *Introduction* section.

6.8 Shibboleth

Contents:

- *Introduction*
- *Installation*
  - *System Requirements*
  - *Install Apache*
  - *Install Shibboleth*
    - *Install Shibboleth DNF Repo*
    - *Install Shibboleth Via DNF*
- *Configure Payara*
  - *App Server HTTP and HTTPS ports*
  - *AJP*
  - *SSLEngine Warning Workaround*
- *Configure Apache*
  - *Enforce HTTPS*
  - *Edit Apache ssl.conf File*
- *Configure Shibboleth*
  - *shibboleth2.xml*
    - *Specific Identity Provider(s)*
    - *Identity Federation*
  - *Shibboleth Attributes*
  - *attribute-map.xml*

  - *Shibboleth and ADFS*
- *Disable or Reconfigure SELinux*
  - *Disable SELinux*
  - *Reconfigure SELinux to Accommodate Shibboleth*
- *Restart Apache and Shibboleth*
- *Configure Apache and shibd to Start at Boot*
- *Verify the Metadata URL*
- *If Your Instance is Using Discofeed: Verify DiscoFeed URL*
- *Add the Shibboleth Authentication Provider to Your Dataverse Installation*
- *Exchange Metadata with Your Identity Provider*
- *Backup sp-cert.pem and sp-key.pem Files*
- *Debugging*
- *Converting Accounts*
  - *Converting Local Users to Shibboleth*
  - *Converting Shibboleth Users to Local*
- *Institution-Wide Shibboleth Groups*
- *Multi-Factor Authentication*

6.8.1 Introduction

By configuring and enabling Shibboleth support in your Dataverse installation, your users will be able to log in using the identity system managed by their institution ("single sign on", or at least "single password") rather than having to create yet another password local to your Dataverse installation. Typically, users know their login system by some sort of internal branding such as "HarvardKey" or "Touchstone" (MIT) but within the Dataverse Software application, the Shibboleth feature is known as *Institutional Log In* as explained to end users in the *Account Creation + Management* section of the User Guide.

Shibboleth is an implementation of the Security Assertion Markup Language (SAML) protocol which is similar in spirit to systems used by many webapps that allow you to log in via Google, Facebook, or Twitter.

Shibboleth can be compared and contrasted with OAuth2, which you can read about in the *OAuth Login Options* section.

6.8.2 Installation

We assume you've already gone through a basic installation as described in the *Installation* section and that you've paid particular attention to the *Auth Modes: Local vs. Remote vs. Both* explanation in the *Configuration* section. You're going to give Shibboleth a whirl. Let's get started.

System Requirements

Support for Shibboleth in the Dataverse Software is built on the popular "mod_shib" Apache module, "shibd" daemon, and the Embedded Discovery Service (EDS) Javascript library, all of which are distributed by the Shibboleth Consortium. EDS is bundled with the Dataverse Software, but `mod_shib` and `shibd` must be installed and configured per below.

Only Red Hat Enterprise Linux (RHEL) and derivatives have been tested (x86_64 versions) by the Dataverse Project team. See https://shibboleth.atlassian.net/wiki/spaces/SP3/pages/2065335547/LinuxInstall for details and note that (according to that page) as of this writing Ubuntu and Debian are not officially supported by the Shibboleth project.

Install Apache

We will be "fronting" the app server with Apache so that we can make use of the `mod_shib` Apache module. We will also make use of the `mod_proxy_ajp` module built in to Apache.

We include the `mod_ssl` package to enforce HTTPS per below.

```
dnf install httpd mod_ssl
```

Install Shibboleth

Installing Shibboleth will give us both the `shibd` service and the `mod_shib` Apache module.

Install Shibboleth DNF Repo

The Shibboleth project now provides a web form to generate an appropriate package repository for use with DNF.

You'll want to copy-paste the form results into `/etc/yum.repos.d/shibboleth.repo` or wherever is most appropriate for your operating system.

Install Shibboleth Via DNF

Please note that during the installation it's ok to import GPG keys from the Shibboleth project. We trust them.

```
dnf install shibboleth
```

6.8.3 Configure Payara

App Server HTTP and HTTPS ports

Apache will be listening on ports 80 and 443 so we need to make sure the app server isn't using them. If you've been changing the default ports used by the app server per the *Configuration* section, revert the HTTP service to listen on 8080, the default port:

```
./asadmin set server-config.network-config.network-listeners.network-listener.
http-listener-1.port=8080
```

Likewise, if necessary, revert the HTTPS service to listen on port 8181:

```
./asadmin set server-config.network-config.network-listeners.network-listener.
http-listener-2.port=8181
```

AJP

A `jk-connector` network listener should have already been set up when you ran the installer mentioned in the *Installation* section, but for reference, here is the command that is used:

```
./asadmin create-network-listener --protocol http-listener-1 --listenerport 8009
--jkenabled true jk-connector
```

You can verify this with `./asadmin list-network-listeners`.

This enables the AJP protocol used in Apache configuration files below.

SSLEngine Warning Workaround

This workaround was required for Glassfish 4 but it is unknown if it is required under Payara.

When fronting Payara with Apache and using the jk-connector (AJP, mod_proxy_ajp), in your Payara server.log you can expect to see "WARNING … org.glassfish.grizzly.http.server.util.RequestUtils … jk-connector … Unable to populate SSL attributes java.lang.IllegalStateException: SSLEngine is null".

To hide these warnings, run `./asadmin set-log-levels org.glassfish.grizzly.http.server.util.RequestUtils=SEVERE` so that the WARNING level is hidden as recommended at https://java.net/jira/browse/GLASSFISH-20694 and https://github.com/IQSS/dataverse/issues/643#issuecomment-49654847

6.8.4 Configure Apache

Enforce HTTPS

To prevent attacks such as FireSheep, HTTPS should be enforced. https://wiki.apache.org/httpd/RewriteHTTPToHTTPS provides a good method. You **could** copy and paste that those "rewrite rule" lines into Apache's main config file at `/etc/httpd/conf/httpd.conf` but using Apache's "virtual hosts" feature is recommended so that you can leave the main configuration file alone and drop a host-specific file into place.

Below is an example of how "rewrite rule" lines look within a `VirtualHost` block. Download a `sample file` , edit it to substitute your own hostname under `ServerName`, and place it at `/etc/httpd/conf.d/dataverse.example.edu.conf` or a filename that matches your hostname. The file must be in `/etc/httpd/conf.d` and must end in ".conf" to be included in Apache's configuration.

```
<VirtualHost *:80>

ServerName dataverse.example.edu

# From https://wiki.apache.org/httpd/RewriteHTTPToHTTPS

RewriteEngine On
# This will enable the Rewrite capabilities

RewriteCond %{HTTPS} !=on
# This checks to make sure the connection is not already HTTPS

RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R,L]
# This rule will redirect users from their original location, to the same location but␣
↪using HTTPS.
# i.e.  http://www.example.com/foo/ to https://www.example.com/foo/
# The leading slash is made optional so that this will work either in httpd.conf
```

(continues on next page)

(continued from previous page)

```
# or .htaccess context

</VirtualHost>
```

Edit Apache ssl.conf File

`/etc/httpd/conf.d/ssl.conf` should be edited to contain the FQDN of your hostname like this: `ServerName dataverse.example.edu:443` (substituting your hostname).

Near the bottom of `/etc/httpd/conf.d/ssl.conf` but before the closing `</VirtualHost>` directive, add the following:

```
# don't pass paths used by Shibboleth to Payara
ProxyPassMatch ^/Shibboleth.sso !
ProxyPassMatch ^/shibboleth-ds !
# pass everything else to Payara
ProxyPass / ajp://localhost:8009/

<Location /shib.xhtml>
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  require valid-user
</Location>
```

You can download a `sample ssl.conf file` to compare it against the file you edited.

Note that `/etc/httpd/conf.d/shib.conf` and `/etc/httpd/conf.d/shibboleth-ds.conf` are expected to be present from installing Shibboleth via dnf.

You may wish to also add a timeout directive to the ProxyPass line within ssl.conf. This is especially useful for larger file uploads as apache may prematurely kill the connection before the upload is processed.

e.g. `ProxyPass / ajp://localhost:8009/ timeout=600` defines a timeout of 600 seconds.

Try to strike a balance with the timeout setting. Again a timeout too low will impact file uploads. A timeout too high may cause additional stress on the server as it will have to service idle clients for a longer period of time.

6.8.5 Configure Shibboleth

shibboleth2.xml

`/etc/shibboleth/shibboleth2.xml` should look something like the `sample shibboleth2.xml file` below, but you must substitute your hostname in the `entityID` value. If your starting point is a `shibboleth2.xml` file provided by someone else, you must ensure that `attributePrefix="AJP_"` is added under `ApplicationDefaults` per the Shibboleth wiki. Without the `AJP_` configuration in place, the required *Shibboleth Attributes* will be null and users will be unable to log in.

```
<!--
This is an example shibboleth2.xml generated originally by http://testshib.org
and tweaked for Dataverse.  See also:

- attribute-map.xml
- dataverse-idp-metadata.xml
```

(continues on next page)

(continued from previous page)

```
https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPConfiguration
-->

<SPConfig xmlns="urn:mace:shibboleth:3.0:native:sp:config" xmlns:md=
↪"urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    clockSkew="1800">

    <!-- FIXME: change the entityID to your hostname. -->
    <ApplicationDefaults entityID="https://dataverse.example.edu/sp"
        REMOTE_USER="eppn" attributePrefix="AJP_">

        <!-- You should use secure cookies if at all possible.  See cookieProps in this␣
↪Wiki article. -->
        <!-- https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPSessions -->
        <Sessions lifetime="28800" timeout="3600" checkAddress="false" relayState="ss:mem
↪" handlerSSL="false" redirectLimit="exact">

            <SSO>
              SAML2 SAML1
            </SSO>
            <!-- If you are planning to use Shibboleth as a member of the InCommon␣
↪federation, -->
            <!-- comment out the section above, and un-comment out the following, in␣
↪order to -->
            <!-- use the new WayFinder service for the login page workflow: -->
            <!-- SSO discoveryProtocol="SAMLDS" discoveryURL="https://wayf.
↪incommonfederation.org/DS/WAYF">
              SAML2 SAML1
            </SSO -->

            <!-- SAML and local-only logout. -->
            <!-- https://wiki.shibboleth.net/confluence/display/SHIB2/
↪NativeSPServiceLogout -->
            <Logout>SAML2 Local</Logout>

            <!--
                Handlers allow you to interact with the SP and gather more information. ␣
↪Try them out!
                Attribute values received by the SP through SAML will be visible at:
                http://dataverse.example.edu/Shibboleth.sso/Session
            -->

            <!-- Extension service that generates "approximate" metadata based on SP␣
↪configuration. -->
            <Handler type="MetadataGenerator" Location="/Metadata" signing="false"/>

            <!-- Status reporting service. -->
            <Handler type="Status" Location="/Status" acl="127.0.0.1"/>

            <!-- Session diagnostic service. -->
```

(continues on next page)

(continued from previous page)

```
            <!-- showAttributeValues must be set to true to see attributes at /
↪Shibboleth.sso/Session . -->
            <Handler type="Session" Location="/Session" showAttributeValues="true"/>

            <!-- JSON feed of discovery information. -->
            <Handler type="DiscoveryFeed" Location="/DiscoFeed"/>

        </Sessions>

        <!-- Error pages to display to yourself if something goes horribly wrong. -->
        <Errors supportContact="root@localhost" logoLocation="/shibboleth-sp/logo.jpg"
                styleSheet="/shibboleth-sp/main.css"/>

        <!-- Loads and trusts a metadata file that describes only the Testshib IdP and␣
↪how to communicate with it. -->
        <!-- IdPs we want allow go in /etc/shibboleth/dataverse-idp-metadata.xml -->
        <MetadataProvider type="XML" path="dataverse-idp-metadata.xml" backingFilePath=
↪"local-idp-metadata.xml" legacyOrgNames="true" reloadInterval="7200"/>
        <!-- Uncomment to enable all the Research & Scholarship IdPs from InCommon -->
        <!--
            <MetadataProvider type="MDQ" id="incommon" ignoreTransport="true"␣
↪cacheDirectory="inc-mdq-cache" maxCacheDuration="86400" minCacheDuration="60" baseUrl=
↪"https://mdq.incommon.org/">
            <MetadataFilter type="Signature" certificate="inc-md-cert-mdq.pem"/>
            <MetadataFilter type="RequireValidUntil" maxValidityInterval="1209600"/>
        </MetadataProvider>
        -->

        <!-- Attribute and trust options you shouldn't need to change. -->
        <AttributeExtractor type="XML" validate="true" path="attribute-map.xml"/>
        <AttributeResolver type="Query" subjectMatch="true"/>
        <AttributeFilter type="XML" validate="true" path="attribute-policy.xml"/>

        <!-- Your SP generated these credentials.  They're used to talk to IdP's. -->
        <CredentialResolver type="File" key="sp-key.pem" certificate="sp-cert.pem"/>

    </ApplicationDefaults>

    <!-- Security policies you shouldn't change unless you know what you're doing. -->
    <SecurityPolicyProvider type="XML" validate="true" path="security-policy.xml"/>

    <!-- Low-level configuration about protocols and bindings available for use. -->
    <ProtocolProvider type="XML" validate="true" reloadChanges="false" path="protocols.
↪xml"/>

</SPConfig>
```

Specific Identity Provider(s)

When configuring the `MetadataProvider` section of `shibboleth2.xml` you should consider if your users will all come from the same Identity Provider (IdP) or not.

Most Dataverse installations will probably only want to authenticate users via Shibboleth using their home institution's Identity Provider (IdP). The configuration above in `shibboleth2.xml` looks for the metadata for the Identity Providers (IdPs) in a file at `/etc/shibboleth/dataverse-idp-metadata.xml`. You can download a `sample dataverse-idp-metadata.xml file` and that includes the SAMLtest IdP from https://samltest.id but you will want to edit this file to include the metadata from the Identity Provider you care about. The identity people at your institution will be able to provide you with this metadata and they will very likely ask for a list of attributes that the Dataverse Software requires, which are listed at *Shibboleth Attributes*.

Identity Federation

Rather than or in addition to specifying individual Identity Providers (see *Specific Identity Provider(s)* above) you may wish to broaden the number of users who can log into your Dataverse installation (to include collaborators, for example) by registering it as a Service Provider (SP) within an identity federation.

For example, in the United States, you would register your Dataverse installation with InCommon. For a list of federations around the world, see REFEDS (the Research and Education FEDerations group). The details of how to register with an identity federation are out of scope for this document.

If you are planning to use InCommon, please note that `shibd` needs to be configured to use the new MDQ protocol and WayFinder service announced by InCommon. The sample `shibboleth2.xml` provided already contains commented-out sections pre-configured to work with this new InCommon framework. Please see https://spaces.at.internet2.edu/display/MDQ/how-to-configure-shib-sp-to-use-mdq and https://spaces.at.internet2.edu/display/federation/how-to-configure-service-to-use-wayfinder for more information. You will also need to set the *dataverse.feature.shibboleth-use-wayfinder* feature flag to true.

For a successful login to Dataverse, certain *Shibboleth Attributes* must be released by the Identity Provider (IdP). Otherwise, in the federation context, users will have the frustrating experience of selecting their IdP in the list but then getting an error like `Problem with Identity Provider - The SAML assertion for "eppn" was null`. We definitely want to prevent this! There's even some guidance about this problem in the User Guide under the heading *Troubleshooting Federated Institutional Log In* that links back here.

For InCommon, a decent strategy for ensuring that IdPs release the necessary attributes is to have both the SP (your Dataverse installation) and the IdP (there are many of these around the world) join the Research & Scholarship (R&S) category. The R&S website explains the R&S dream well:

> "The Research and Scholarship (R&S) entity category defines a simple and scalable way to streamline federated research access. Identity providers (IdP) supporting R&S category agree to release basic, predefined person directory information to all service providers (SP) serving the Research and Scholarship community."

In short, R&S IdPs trust R&S SPs and vice versa. R&S SPs (like Dataverse) agree to only require attributes that R&S IdPs agree to release (the "Research & Scholarship Attribute Bundle"). Ideally, there is no need to make special arrangements with each IdP.

For InCommon, follow their instructions to make your Dataverse installation an R&S SP. For other federations, consult their documentation.

Unfortunately, in practice, some R&S IdPs do not release the attributes they agreed to release to R&S SPs when joining R&S. In this case, you will have to contact the IdP, show them the list of *Shibboleth Attributes* that Dataverse requires for a successful login, and try to convince them to release them.

Shibboleth Attributes

The following attributes are required for a successful Shibboleth login:

- Shib-Identity-Provider
- eppn
- givenName
- sn
- email

See also https://incommon.org/federation/attributes/ and https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPAttributeAccess

attribute-map.xml

By default, some attributes `/etc/shibboleth/attribute-map.xml` are commented out and "subject-id" is used instead of "eppn". We recommend downloading and using `attribute-map.xml` instead which has these changes and should be compatible with the Dataverse installation.

Shibboleth and ADFS

With appropriate configuration, your Dataverse installation and Shibboleth can make use of "single sign on" using Active Directory. This requires configuring `shibd` and `httpd` to load appropriate libraries, and insuring that the attribute mapping matches those provided. Example configuration files for `shibboleth2.xml` and `attribute-map.xml` may be helpful. Note that your ADFS server hostname goes in the file referenced under "MetadataProvider" in your shibboleth2.xml file.

6.8.6 Disable or Reconfigure SELinux

SELinux is set to "enforcing" by default on RHEL/CentOS, but unfortunately Shibboleth does not "just work" with SELinux. You have two options. You can disable SELinux or you can reconfigure SELinux to accommodate Shibboleth.

Disable SELinux

The first and easiest option is to set `SELINUX=permisive` in `/etc/selinux/config` and run `setenforce permissive` or otherwise disable SELinux to get Shibboleth to work. This is apparently what the Shibboleth project expects because their wiki page says, "At the present time, we do not support the SP in conjunction with SELinux, and at minimum we know that communication between the mod_shib and shibd components will fail if it's enabled. Other problems may also occur."

Reconfigure SELinux to Accommodate Shibboleth

Issue the following commands to allow Shibboleth to function when SELinux is enabled:

```
# Allow httpd to connect to network and read content
sudo /usr/sbin/setsebool -P httpd_can_network_connect 1
sudo /usr/sbin/setsebool -P httpd_read_user_content 1

# Allow httpd to connect to Shib socket
sudo grep httpd_t /var/log/audit/audit.log |/usr/bin/audit2allow -M allow_httpd_shibd_
↪sock
sudo /usr/sbin/semodule -i allow_httpd_shibd_sock.pp

# Allow httpd to read /var/cache/shibboleth
sudo /usr/sbin/semanage fcontext -a -t httpd_sys_content_t "/var/cache/shibboleth(/.*)?"
sudo /usr/sbin/restorecon -vR /var/cache/shibboleth
```

6.8.7 Restart Apache and Shibboleth

After configuration is complete, restart `shib` and `httpd`.

On CentOS 7:

```
systemctl restart shibd.service
```

```
systemctl restart httpd.service
```

On CentOS 6:

```
service shibd restart
```

```
service httpd restart
```

6.8.8 Configure Apache and shibd to Start at Boot

On CentOS 7/8:

```
systemctl enable httpd.service
```

```
systemctl enable shibd.service
```

On CentOS 6:

```
chkconfig httpd on
```

```
chkconfig shibd on
```

6.8.9 Verify the Metadata URL

Substitute your hostname and verify that you are seeing your service provider metadata in XML format:

- https://dataverse.example.edu/Shibboleth.sso/Metadata

6.8.10 If Your Instance is Using Discofeed: Verify DiscoFeed URL

As another sanity check, substitute your hostname and make sure you see well-formed JSON:

- https://dataverse.example.edu/Shibboleth.sso/DiscoFeed

(Skip this step if you'll be using Shibboleth as a registered member of InCommon federation, since the DiscoFeed will not be part of the workflow.)

The JSON in `DiscoFeed` comes from the list of IdPs you configured in the `MetadataProvider` section of `shibboleth2.xml` and will form a dropdown list on the Login Page.

6.8.11 Add the Shibboleth Authentication Provider to Your Dataverse Installation

Now that you've configured your app server, Apache, and `shibd`, you are ready to turn your attention back to the Dataverse installation to enable Shibboleth as an "authentication provider." You will be using `curl` to POST the following JSON file to the `authenticationProviders` endpoint of the *Native API*.

```
{
    "id":"shib",
    "factoryAlias":"shib",
    "enabled":true
}
```

```
curl -X POST -H 'Content-type:  application/json' --upload-file shibAuthProvider.json
http://localhost:8080/api/admin/authenticationProviders
```

Now that you've added the Shibboleth authentication provider to your Dataverse installation, as described in the *Account Creation + Management* section of the User Guide, you should see a new "Your Institution" button under "Other Log In Options" on the Log In page. After clicking "Your Institution", you should see the institutions you configured in `/etc/shibboleth/shibboleth2.xml` above. If not, double check the content of the `DiscoFeed` URL above. If you don't see the "Your Institution" button, confirm that the the "shib" authentication provider has been added by listing all the authentication providers the Dataverse installation knows about:

```
curl http://localhost:8080/api/admin/authenticationProviders
```

Once you have confirmed that the Dataverse installation's web interface is listing the institutions you expect, you'll want to temporarily remove the Shibboleth authentication provider you just added because users won't be able to log in via their institution until you have exchanged metadata with one or more Identity Providers (IdPs), which is described below. As explained in the section of the *Native API* of the API Guide, you can delete an authentication provider by passing its `id`:

```
curl -X DELETE http://localhost:8080/api/admin/authenticationProviders/shib
```

Before contacting your actual Identity Provider, we recommend testing first with the "SAMLtest" Identity Provider (IdP) to ensure that you have configured everything correctly. This process is described next.

6.8.12 Exchange Metadata with Your Identity Provider

https://samltest.id (SAMLtest) is a fantastic resource for testing Shibboleth configurations. Depending on your relationship with your identity people you may want to avoid bothering them until you have tested your Dataverse installation configuration with the SAMLtest Identity Provider (IdP). This process is explained below.

If you've temporarily configured your `MetadataProvider` to use the SAMLtest Identity Provider (IdP) as outlined above, you can download your metadata like this (substituting your hostname in both places):

```
curl https://dataverse.example.edu/Shibboleth.sso/Metadata > dataverse.example.edu
```

Then upload your metadata to https://samltest.id/upload.php (or click "Fetch").

Then try to log in to your Dataverse installation using the SAMLtest IdP. After logging in, you can visit the https://dataverse.example.edu/Shibboleth.sso/Session (substituting your hostname) to troubleshoot which attributes are being received. You should see something like the following:

```
Miscellaneous
Session Expiration (barring inactivity): 479 minute(s)
Client Address: 75.69.182.6
SSO Protocol: urn:oasis:names:tc:SAML:2.0:protocol
Identity Provider: https://samltest.id/saml/idp
Authentication Time: 2019-11-28T01:23:28.381Z
Authentication Context Class: urn:oasis:names:tc:SAML:2.
↪0:ac:classes:PasswordProtectedTransport
Authentication Context Decl: (none)

Attributes
displayName: Rick Sanchez
eppn: rsanchez@samltest.id
givenName: Rick
mail: rsanchez@samltest.id
sn: Sanchez
telephoneNumber: +1-555-555-5515
uid: rick
```

When you are done testing, you can delete the SAMLtest users you created like this (after you have deleted any data and permisions associated with the users):

```
curl -X DELETE http://localhost:8080/api/admin/authenticatedUsers/rick
```

(Of course, you are also welcome to do a fresh reinstall per the *Installation* section.)

If your Dataverse installation is working with SAMLtest it **should** work with your institution's Identity Provider (IdP). Next, you should:

- Send your identity people your metadata file above (or a link to download it themselves). From their perspective you are a Service Provider (SP).
- Ask your identity people to send you the metadata for the Identity Provider (IdP) they operate. See the section above on `shibboleth2.xml` and `MetadataProvider` for what to do with the IdP metadata. Restart `shibd` and `httpd` as necessary.
- Re-add Shibboleth as an authentication provider to your Dataverse installation as described above.
- Test login to your Dataverse installation via your institution's Identity Provider (IdP).

6.8.13 Backup sp-cert.pem and sp-key.pem Files

Especially if you have gotten authentication working with your institution's Identity Provider (IdP), now is the time to make sure you have backups.

The installation and configuration of Shibboleth will result in the following cert and key files being created and it's important to back them up. The cert is in the metadata you shared with your IdP:

- `/etc/shibboleth/sp-cert.pem`
- `/etc/shibboleth/sp-key.pem`

If you have more than one Payara server, you should use the same `sp-cert.pem` and `sp-key.pem` files on all of them. If these files are compromised and you need to regenerate them, you can `cd /etc/shibboleth` and run `keygen.sh` like this (substituting you own hostname):

```
./keygen.sh -f -u shibd -g shibd -h dataverse.example.edu -e https://dataverse.example.
edu/sp
```

6.8.14 Debugging

The *Troubleshooting* section of the Admin Guide explains how to increase Payara logging levels. The relevant classes and packages are:

- edu.harvard.iq.dataverse.Shib
- edu.harvard.iq.dataverse.authorization.providers.shib
- edu.harvard.iq.dataverse.authorization.groups.impl.shib

6.8.15 Converting Accounts

As explained in the *Account Creation + Management* section of the User Guide, users can convert from one login option to another.

Converting Local Users to Shibboleth

If you are running in "remote and local" mode and have existing local users that you'd like to convert to Shibboleth users, give them the following steps to follow, which are also explained in the *Account Creation + Management* section of the User Guide:

- Log in with your local account to make sure you know your password, which will be needed for the account conversion process.
- Log out of your local account.
- Log in with your Shibboleth account.
- If the email address associated with your local account matches the email address asserted by the Identity Provider (IdP), you will be prompted for the password of your local account and asked to confirm the conversion of your account. You're done! Browse around to ensure you see all the data you expect to see. Permissions have been preserved.
- If the email address asserted by the Identity Provider (IdP) does not match the email address of any local user, you will be prompted to create a new account. If you were expecting account conversion, you should decline creating a new Shibboleth account, log back in to your local account, and let Support know the email on file for your local account. Support may ask you to change your email address for your local account to the one that is being asserted by the Identity Provider. Someone with access to the Payara logs will see this email address there.

Converting Shibboleth Users to Local

Whereas users convert their own accounts from local to Shibboleth as described above, conversion in the opposite direction is performed by a sysadmin. A common scenario may be as follows:

- A user emails Support saying, "I left the university (or wherever) and can't log in to the Dataverse installation anymore. What should I do?"
- Support replies asking the user for a new email address (Gmail, new institution email, etc.) to associate with their Dataverse installation account.
- The user replies with a new email address to associate with their Dataverse installation account.
- Support runs the curl command below, supplying the database id of the user to convert and the new email address and notes the username returned.
- Support emails the user and indicates that that they should use the password reset feature to set a new password and to make sure to take note of their username under Account Information (or the password reset confirmation email) since the user never had a username before.
- The user resets password and is able to log in with their local account. All permissions have been preserved with the exception of any permissions assigned to an institution-wide Shibboleth group to which the user formerly belonged.

In the example below, the user has indicated that the new email address they'd like to have associated with their account is "former.shib.user@mailinator.com" and their user id from the `authenticateduser` database table is "2". The API token must belong to a superuser (probably the sysadmin executing the command). Note that the old version of this call, *convertShibToBuiltIn*, is deprecated and will be deleted in a future release.

```
curl -H "X-Dataverse-key:  $API_TOKEN" -X PUT -d "former.shib.user@mailinator.com" http:/
/localhost:8080/api/admin/authenticatedUsers/id/2/convertRemoteToBuiltIn
```

Rather than looking up the user's id in the `authenticateduser` database table, you can issue this command to get a listing of all users:

```
curl -H "X-Dataverse-key:  $API_TOKEN" http://localhost:8080/api/admin/authenticatedUsers
```

Per above, you now need to tell the user to use the password reset feature to set a password for their local account.

6.8.16 Institution-Wide Shibboleth Groups

The Dataverse Software allows you to optionally define "institution-wide Shibboleth groups" based on the the entityID of the Identity Provider (IdP) used to authenticate. For example, an "institution-wide Shibboleth group" with `https://samltest.id/saml/idp` as the IdP would include everyone who logs in via the SAMLtest IdP mentioned above.

To create an institution-wide Shibboleth groups, create a JSON file like `shibGroupSAMLtest.json` as below and issue this curl command:

```
curl http://localhost:8080/api/admin/groups/shib -X POST -H 'Content-type:application/
json' --upload-file shibGroupSAMLtest.json
```

```
{
  "name": "All samltest.id Shibboleth Users",
  "attribute": "Shib-Identity-Provider",
  "pattern": "https://samltest.id/saml/idp"
}
```

Institution-wide Shibboleth groups are based on the "Shib-Identity-Provider" SAML attribute asserted at runtime after successful authentication with the Identity Provider (IdP) and held within the browser session rather than being persisted

in the database for any length of time. It is for this reason that roles based on these groups, such as the ability to create a dataset, are not honored by non-browser interactions, such as through the SWORD API.

To list institution-wide Shibboleth groups: `curl http://localhost:8080/api/admin/groups/shib`

To delete an institution-wide Shibboleth group (assuming id 1): `curl -X DELETE http://localhost:8080/api/admin/groups/shib/1`

Support for arbitrary attributes beyond "Shib-Identity-Provider" such as "eduPersonScopedAffiliation", etc. is being tracked at https://github.com/IQSS/dataverse/issues/1515

6.8.17 Multi-Factor Authentication

Institutions that wish to require MFA for their own accounts may add

```
authnContextClassRef="https://refeds.org/profile/mfa"
```

to the shibboleth2.xml SSO element.

Federated institutions that would like to require MFA for their own account but not require MFA of other federated institutions may add

```
<RelyingParty Name="urn:mace:incommon:yourinstitution.edu" authnContextClassRef="https://
↪refeds.org/profile/mfa"/>
```

to shibboleth2.xml, beneath the Sessions and Errors elements.

6.9 OAuth Login Options

Contents:

- *Introduction*
- *Setup*
  - *Identity Provider Side*
    - *Obtain Client ID and Client Secret*
  - *Dataverse Installation Side*
    - *ORCID Sandbox*
  - *Disabling Sign Up*
- *Converting Local Users to OAuth*
- *Converting OAuth Users to Local*

6.9.1 Introduction

As explained under "Auth Modes" in the *Configuration* section, OAuth2 is one of the ways that you can have end users log in to your Dataverse installation.

OAuth2 is an authentication protocol that allows systems to share user data, while letting the users control what data is being shared. When you see buttons stating "login with Google" or "login through Facebook", OAuth2 is probably involved. For the purposes of this section, we will shorten "OAuth2" to just "OAuth." OAuth can be compared and contrasted with *Shibboleth*.

The Dataverse Software supports four OAuth providers: ORCID, Microsoft Azure Active Directory (AD), GitHub, and Google.

In addition *OpenID Connect Login Options* are supported, using a standard based on OAuth2.

6.9.2 Setup

Setting up an OAuth identity provider to work with your Dataverse installation requires setup in two places: the provider, and the Dataverse installation.

Identity Provider Side

Obtain Client ID and Client Secret

Before OAuth providers will release information about their users (first name, last name, etc.) to your Dataverse installation, you must request a "Client ID" and "Client Secret" from them. In many cases you can use providers' automated system to request these credentials, but if not, contact the provider for assistance.

URLs to help you request a Client ID and Client Secret from the providers supported by the Dataverse Software are provided below. For all of these providers, it's a good idea to request the Client ID and Client secret using a generic account, perhaps the one that's associated with the `:SystemEmail` you've configured for your Dataverse installation, rather than your own personal Microsoft Azure AD, ORCID, GitHub, or Google account:

- ORCID: https://orcid.org/content/register-client-application-0
- Microsoft: https://docs.microsoft.com/en-us/azure/active-directory/develop/v1-protocols-oauth-code
- GitHub: https://github.com/settings/applications/new via https://developer.github.com/v3/oauth/
- Google: https://console.developers.google.com/projectselector/apis/credentials via https://developers.google.com/identity/protocols/OAuth2WebServer (pick "OAuth client ID")

Each of these providers will require the following information from you:

- Basic information about your Dataverse installation such as a name, description, URL, logo, privacy policy, etc.
- OAuth2 Redirect URI(s) (ORCID) or Redirect URI (Microsoft Azure AD) or Authorization Callback URL (GitHub) or Authorized Redirect URIs (Google): This is the URL on the Dataverse installation side to which the user will be sent after successfully authenticating with the identity provider. This should be the advertised URL of your Dataverse installation (the protocol, fully qualified domain name, and optional port configured via the `dataverse.siteUrl` JVM option mentioned in the *Configuration* section) appended with `/oauth2/callback.xhtml` such as `https://dataverse.example.edu/oauth2/callback.xhtml`.

For ORCID, if you also want to enable the ability to associate ORCIDs with user accounts (when users did not login via ORCID) as discussed in *ORCID Integration*, you must add a second redirect URL: to `/oauth2/orcidConfirm.xhtml` such as `https://dataverse.example.edu/oauth2/orcidConfirm.xhtml`.

When you are finished you should have a Client ID and Client Secret from the provider. Keep them safe and secret.

Dataverse Installation Side

As explained under "Auth Modes" in the *Configuration* section, available authentication providers are stored in the `authenticationproviderrow` database table and can be listed with this command:

```
curl http://localhost:8080/api/admin/authenticationProviders
```

We will `POST` a JSON file containing the Client ID and Client Secret to this `authenticationProviders` API endpoint to add another authentication provider. As a starting point, you'll want to download the JSON template file matching the provider you're setting up:

- `orcid-public.json`
- `orcid-member.json`
- `github.json`
- `google.json`
- `microsoft.json`

Here's how the JSON template for GitHub looks, for example:

```
{
    "id":"github",
    "factoryAlias":"oauth2",
    "title":"GitHub",
    "subtitle":"",
    "factoryData":"type: github | userEndpoint: NONE | clientId: FIXME | clientSecret:␣
↪FIXME",
    "enabled":true
}
```

Edit the JSON template and replace the two "FIXME" values with the Client ID and Client Secret you obtained earlier. Then use curl to `POST` the JSON to the Dataverse installation:

```
curl -X POST -H 'Content-type:  application/json' --upload-file github.json http://
localhost:8080/api/admin/authenticationProviders
```

After restarting your app server you should see the new provider under "Other options" on the Log In page, as described in the *Account Creation + Management* section of the User Guide.

By default, the Log In page will show the "builtin" provider, but you can adjust this via the `:DefaultAuthProvider` configuration option. For details, see *Configuration*.

ORCID Sandbox

ORCID provides a sandbox registry, which may be useful for staging, or for development installations. This template can be used for configuring this setting (**this is not something you should use in a production environment**):

- `orcid-sandbox.json`

Disabling Sign Up

See *:AllowRemoteAuthSignUp*.

6.9.3 Converting Local Users to OAuth

Once you have enabled at least one OAuth provider, existing users might want to change their login method from local to OAuth to avoid having a Dataverse installation-specific password. This is documented from the end user perspective in the *Account Creation + Management* section of the User Guide. Users will be prompted to create a new account but can choose to convert an existing local account after confirming their password.

6.9.4 Converting OAuth Users to Local

Whereas users convert their own accounts from local to OAuth as described above, conversion in the opposite direction is performed by a sysadmin. A common scenario may be as follows:

- A user emails Support saying, "Rather than logging in with Google, I want to log in with ORCID (or a local password). What should I do?"
- Support replies asking the user for a new email address to associate with their Dataverse installation account.
- The user replies with a new email address to associate with their Dataverse installation account.
- Support runs the curl command below, supplying the database id of the user to convert and the new email address and notes the username returned.
- Support emails the user and indicates that they should use the password reset feature to set a new password and to make sure to take note of their username under Account Information (or the password reset confirmation email) since the user never had a username before.
- The user resets password and is able to log in with their local account. All permissions have been preserved. The user can continue to log in with this Dataverse installation-specific password or they can convert to an identity provider, if available.

In the example below, the user has indicated that the new email address they'd like to have associated with their account is "former.oauth.user@mailinator.com" and their user id from the `authenticateduser` database table is "42". The API token must belong to a superuser (probably the sysadmin executing the command).

```
curl -H "X-Dataverse-key:  $API_TOKEN" -X PUT -d "former.oauth.user@mailinator.com"
http://localhost:8080/api/admin/authenticatedUsers/id/42/convertRemoteToBuiltIn
```

The expected output is something like this:

```
{
  "status": "OK",
  "data": {
    "email": "former.oauth.user@mailinator.com",
    "username": "jdoe"
  }
}
```

Rather than looking up the user's id in the `authenticateduser` database table, you can issue this command to get a listing of all users:

```
curl -H "X-Dataverse-key:  $API_TOKEN" http://localhost:8080/api/admin/authenticatedUsers
```

Per above, you now need to tell the user to use the password reset feature to set a password for their local account.

6.10 OpenID Connect Login Options

Contents:

- *Introduction*
- *Other Use Cases and Combinations*
- *How to Use*
  - *Enabling PKCE Security*
- *Provision a Provider*
  - *Provision via REST API*
  - *Provision via JVM Options*

6.10.1 Introduction

The OpenID Connect (or OIDC) standard support is closely related to our *OAuth Login Options*, as it has been based on the OAuth 2.0 standard. Quick summary: OIDC is using OAuth 2.0, but adds a standardized way how authentication is done, while this is up to providers when using OAuth 2.0 for authentication.

Being a standard, you can easily enable the use of any OpenID connect compliant provider out there for login into your Dataverse installation.

Some prominent provider examples:

- Google
- Microsoft Azure AD
- Yahoo
- ORCID announced support

You can also either host an OpenID Connect identity management on your own or use a customizable hosted service:

- Okta is a hosted solution
- Keycloak is an open source solution for an IDM/IAM
- Unity IDM is another open source IDM/IAM solution

6.10.2 Other Use Cases and Combinations

- Using your custom identity management solution might be a workaround when you seek for LDAP support, but don't want to go for services like Microsoft Azure AD et al.
- You want to enable users to login in multiple different ways but appear as one account to the Dataverse installation. This is currently not possible within the Dataverse Software itself, but hosting an IDM and attaching the Dataverse installation solves it.
- You want to use the eduGain Federation or other well known SAML federations, but don't want to deploy Shibboleth as your service provider. Using an IDM solution in front easily allows you to use them without hassle.

- There's also a Shibboleth IdP (not SP!) extension, so if you already have a Shibboleth identity provider at your institution, you can reuse it more easily with your Dataverse installation.
- In the future, OpenID Connect might become a successor to the large scale R&E SAML federations we have nowadays. See also OpenID Connect Federation Standard (in development)

6.10.3 How to Use

Just like with *OAuth Login Options* you need to obtain a *Client ID* and a *Client Secret* from your provider(s).

Note

The Dataverse Software does not support OpenID Connect Dynamic Registration. You need to apply for credentials out-of-band.

The Dataverse installation will discover all necessary metadata for a given provider on its own (this is part of the standard).

To enable this, you need to specify an *Issuer URL* when creating the configuration for your provider (see below).

Finding the issuer URL is best done by searching for terms like "discovery" in the documentation of your provider. The discovery document is always located at `<issuer url>/.well-known/openid-configuration` (standardized). To be sure, you can always lookup the `issuer` value inside the live JSON-based discovery document.

Note if you work with Keycloak, make sure the base URL is in the following format: `https://host:port/realms/{realm}` where `{realm}` has to be replaced by the name of the Keycloak realm.

After adding a provider, the Log In page will by default show the "builtin" provider, but you can adjust this via the `:DefaultAuthProvider` configuration option. For details, see *Configuration*.

Hint

In contrast to our *OAuth Login Options*, you can use multiple providers by creating distinct configurations enabled by the same technology and without modifying the Dataverse Software code base (standards for the win!).

Enabling PKCE Security

Many providers these days support or even require the usage of PKCE to safeguard against some attacks and enable public clients that cannot have a secure secret to still use OpenID Connect (or OAuth2).

The Dataverse-built OIDC client can be configured to use PKCE and the method to use when creating the code challenge can be specified. See also this explanation of the flow for details on how this works.

As we are using the Nimbus SDK as our client library, we support the standard `PLAIN` and `S256` (SHA-256) code challenge methods. "SHA-256 method" is the default as recommend in RFC7636. If your provider needs some other method, please open an issue.

The provisioning sections below contain in the example the parameters you may use to configure PKCE.

6.10.4 Provision a Provider

Depending on your use case, you can choose different ways to setup one or multiple OIDC identity providers.

Using *JVM Options* has the advantage of being consistent and does not require additional calls to the API. It can only configure one provider though, yet you can mix with other provider definitions via API.

Using the REST API has the advantage of provisioning multiple, different OIDC providers. Depending on your use case, it has the drawback of needing additional API calls.

If you only need one single provider in your installation and it is using OIDC, use the JVM options, as it requires fewer extra steps and allows you to keep more configuration in a single source.

Provision via REST API

Note: you may omit the PKCE related settings from `factoryData` below if you don't plan on using PKCE - default is disabled.

Please create a `my-oidc-provider.json` file, replacing every `<...>` with your values:

```
{
    "id":"<a unique id>",
    "factoryAlias":"oidc",
    "title":"<a title - shown in UI>",
    "subtitle":"<a subtitle - currently unused in UI>",
    "factoryData":"type: oidc | issuer: <issuer url> | clientId: <client id> |␣
↪clientSecret: <client secret> | pkceEnabled: <true/false> | pkceMethod: <PLAIN/S256/...
↪>",
    "enabled":true
}
```

Now load the configuration into your Dataverse installation using the same API as with *OAuth Login Options*:

```
curl -X POST -H 'Content-type: application/json' --upload-file my-oidc-provider.json
http://localhost:8080/api/admin/authenticationProviders
```

The Dataverse installation will automatically try to load the provider and retrieve the metadata. Watch the app server log for errors. You should see the new provider under "Other options" on the Log In page, as described in the *Account Creation + Management* section of the User Guide.

Provision via JVM Options

A single provider may be provisioned using *JVM Options*. It may be accompanied by more providers configured via REST API. Note that this provider will only be deployed at startup time and (currently) cannot be reconfigured without a restart.

All options below may be set via *MicroProfile Config API* sources. Examples: use environment variable `DATAVERSE_AUTH_OIDC_ENABLED` for the `dataverse.auth.oidc.enabled` option or `DATAVERSE_AUTH_OIDC_CLIENT_ID` for the `dataverse.auth.oidc.client-id` option.

The following options are available:

Option	Description	Mandatory	Default
`dataverse.auth.oidc.enabled`	Enable or disable provisioning the provider via MicroProfile.	N	`false`
`dataverse.auth.oidc.hidden-jsf`	Show or Hide the provider from the JSF UI via MicroProfile.	N	`false`
`dataverse.auth.oidc.client-id`	The client-id of the application to identify it at your provider.	Y	-
`dataverse.auth.oidc.client-secret`	A confidential secret to authorize application requests to the provider as legit.	N	-
`dataverse.auth.oidc.auth-server-url`	The base URL of the OpenID Connect (OIDC) server as explained above.	Y	-
`dataverse.auth.oidc.pkce.enabled`	Set to `true` to enable *PKCE* in auth flow.	N	`false`
`dataverse.auth.oidc.pkce.method`	Set code challenge method. The default value is the current best practice in the literature.	N	`S256`
`dataverse.auth.oidc.title`	The UI visible name for this provider in login options.	N	`OpenID Connect`
`dataverse.auth.oidc.subtitle`	A subtitle, currently not displayed by the UI.	N	`OpenID Connect`
`dataverse.auth.oidc.pkce.max-cache-size`	Tune the maximum size of all OIDC providers' verifier cache (the number of outstanding PKCE-enabled auth responses).	N	10000
`dataverse.auth.oidc.pkce.max-cache-age`	Tune the maximum age, in seconds, of all OIDC providers' verifier cache entries. Default is 5 minutes, equivalent to lifetime of many OIDC access tokens.	N	300

6.11 ORCID Integration

Contents:

- *Introduction*
- *Configuration*

6.11.1 Introduction

Dataverse leverages ORCIDs (and other types of persistent identifiers (PIDs)) to improve the findability of data and to simplify the process of adding metadata. When ORCIDs are included as metadata about authors, Dataverse includes them in metadata exports, advertises them through *Signposting* and via metadata embedded in dataset pages, and includes them in the metadata associated with dataset DOIs.

Dataverse can be configured to make it easier to include ORCIDs - via use of an ORCID "External Vocabulary Script" that allows users to lookup authors, depositors, etc. based on their ORCID profile metadata and then records these ORCIDs automatically and adds links to ORCID profiles in metadata displays. With this configured, there is no need enter ORCIDs directly. See *Using External Vocabulary Services* in the Admin Guide. - via association of ORCIDs with Dataverse user accounts, through the use of ORCID logins or, in addition or instead, a separate authenticated ORCID linking mechanism. When an ORCID is associated with a Dataverse account, it will automatically be added to the dataset metadata when a user creates a dataset and is added as an initial author.

See also *Linking ORCID with Your Account Profile* in the User Guide.

6.11.2 Configuration

The steps needed to configure Dataverse to support lookup of ORCIDs for the author metadata field (and ROR identifiers for organizations as author affiliations) is described in the Dataverse Author Field Example page in the Dataverse External Vocabulary Suport Github Repository. Briefly, this involves changing the *:CVocConf* setting and potentially creating local web-acessible copies of the relevant scripts.

To configure Dataverse to support adding ORCIDs to user profiles, one must configure ORCID as an OAuth2 provider as described in *OAuth Login Options*. The ability to link ORCIDs to user accounts is automatically enabled if an ORCID provider is configured. To avoid also enabling ORCID login, the provider can be registered with "enabled":false.

6.12 External Tools

External tools can provide additional features that are not part of the Dataverse Software itself, such as data exploration.

Contents:

- *Inventory of External Tools*
- *Managing External Tools*
- *Building External Tools*

6.12.1 Inventory of External Tools

See *Inventory of External Tools*.

6.12.2 Managing External Tools

See the *External Tools* section of the Admin Guide.

6.12.3 Building External Tools

See the *Building External Tools* section of the API Guide.

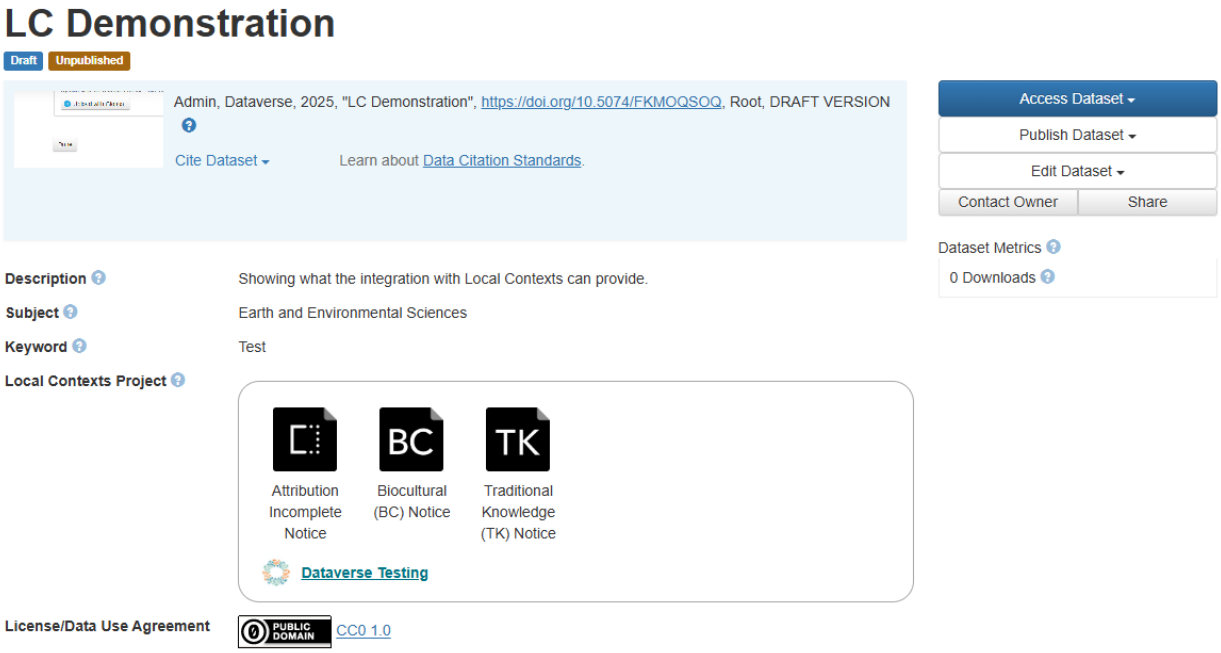
6.13 LocalContexts Integration

Contents:

- *Configuration*

Local Contexts is a global initiative that supports Indigenous communities in the management and sharing of their cultural heritage and data. The Local Contexts Hub is a platform that enables the creation and application of Traditional Knowledge (TK) and Biocultural (BC) Labels and Notices. These labels and notices help to communicate the cultural context and appropriate use of Indigenous data and cultural heritage materials.

Dataverse supports integration with the Local Contexts Hub so that Labels and Notices associated with a dataset can be displayed on the dataset page:



6.13.1 Configuration

There are several steps to LocalContexts integration.

- A Local Contexts Hub Institutional or Integration Partner account is required. See https://localcontexts.org/hub-agreements for more information and the associated costs. (Institutions may wish to connect their Institutional accounts with The Dataverse Project Integration Partner rather than having their own Integration Partner account.) (Free accounts for testing can be created at https://sandbox.localcontextshub.org/.)
- Create an API key in your Local Contexts account.
- Configure the DATAVERSE_LOCALCONTEXTS_URL and DATAVERSE_LOCALCONTEXTS_API_KEY as described in the *dataverse.localcontexts.url* section of the Configuration Guide.
- Add the Local Contexts metadatablock and configure the associated external vocabulary script. Both are available, along with installation instructions, in the Dataverse External Vocabulary GitHub Repository. The metadatablock contains one field allowing Dataverse to store the URL of an associated Local Contexts Hub project. Be sure to update the Solr schema after installing the metadatablock (see *Updating the Solr Schema*). The external vocabulary script interacts with the Local Contexts Hub (via the Dataverse server) to display the Labels and Notices associated with the proect and provide a link to it. The script also supports adding/removing such a link from the dataset's metadata. Note that only a project that references the dataset's PID in its *Optional Project Information* field can be linked to a dataset. Note that the Local Contexts script configuration JSON must be edited to include your Dataverse server's URL and the Local Contexts API key you use in Dataverse. (The latter is optional but it must be included for Dataverse to add information about Notices and Labels to exported metadata and the metadata sent to DataCite for DOIs.)

- Lastly, to show Local Contexts information in the summary section of the dataset page, as shown in the image above, you should add *LCProjectUrl* to list of custom summary fields via use of the *:CustomDatasetSummaryFields* setting.
- Optionally, one can also set the dataverse.feature.add-local-contexts-permission-check FeatureFlag to true. This assures that only users editing datasets can use the LocalContexts search functionality (e.g. via API). This is not recommended unless problematic use is seen.

6.14 Advanced Installation

Advanced installations are not officially supported but here we are at least documenting some tips and tricks that you might find helpful. You can find a diagram of an advanced installation in the *Preparation* section.

Contents:

- *Multiple App Servers*
  - *Detecting Which App Server a User Is On*
- *Licensing*
  - *Standardizing Custom Licenses*
- *Optional Components*
  - *Standalone "Zipper" Service Tool*
  - *External Metadata Exporters*
    - *Enabling External Exporters*
    - *Inventory of External Exporters*
    - *Developing New Exporters*

6.14.1 Multiple App Servers

You should be conscious of the following when running multiple app servers.

- Only one app server can be the dedicated timer server, as explained in the *Dataverse Installation Application Timers* section of the Admin Guide.
- When users upload a logo or footer for their Dataverse collection using the "theme" feature described in the *Dataverse Collection Management* section of the User Guide, these logos are stored only on the app server the user happened to be on when uploading the logo. By default these logos and footers are written to the directory `/usr/local/payara7/glassfish/domains/domain1/docroot/logos`.
- When a sitemap is created by an app server it is written to the filesystem of just that app server. By default the sitemap is written to the directory `/usr/local/payara7/glassfish/domains/domain1/docroot/sitemap`.
- If Make Data Count is used, its raw logs must be copied from each app server to single instance of Counter Processor. See also *:MDCLogPath* section in the Configuration section of this guide and the *Make Data Count* section of the Admin Guide.

- Dataset draft version logging occurs separately on each app server. See *Edit Draft Versions Logging* section in Monitoring of the Admin Guide for details.
- Password aliases (`dataverse.db.password`, etc.) are stored per app server.
- Each app server generates its own signing secret at startup unless one is configured, so a signed URL created on one server cannot be validated on another (and no signed URL survives a restart). Set *dataverse.api.signing-secret* to the same value (at least 36 characters) on every app server if you use signed URLs.

Detecting Which App Server a User Is On

If you have successfully installed multiple app servers behind a load balancer you might like to know which server a user has landed on. A straightforward solution is to place a file called `host.txt` in a directory that is served up by Apache such as `/var/www/html` and then configure Apache not to proxy requests to `/host.txt` to the app server. Here are some example commands on RHEL/derivatives that accomplish this:

```
[root@server1 ~]# vim /etc/httpd/conf.d/ssl.conf
[root@server1 ~]# grep host.txt /etc/httpd/conf.d/ssl.conf
ProxyPassMatch ^/host.txt !
[root@server1 ~]# systemctl restart httpd.service
[root@server1 ~]# echo $HOSTNAME > /var/www/html/host.txt
[root@server1 ~]# curl https://dataverse.example.edu/host.txt
server1.example.edu
```

You would repeat the steps above for all of your app servers. If users seem to be having a problem with a particular server, you can ask them to visit https://dataverse.example.edu/host.txt and let you know what they see there (e.g. "server1.example.edu") to help you know which server to troubleshoot.

Please note that *Network Ports* under the Configuration section has more information on fronting your app server with Apache. The *Shibboleth* section talks about the use of `ProxyPassMatch`.

6.14.2 Licensing

Dataverse allows superusers to specify the list of allowed licenses, to define which license is the default, to decide whether users can instead define custom terms, and to mark obsolete licenses as "inactive" to stop further use of them. These can be accomplished using the *native API* and the *:AllowCustomTermsOfUse* setting. See also *Configuring Licenses*.

Standardizing Custom Licenses

In addition, if many datasets use the same set of Custom Terms, it may make sense to create and register a standard license including those terms. Doing this would include:

- Creating and posting an external document that includes the custom terms, i.e. an HTML document with sections corresponding to the terms fields that are used.
- Defining a name, short description, URL (where it is posted), and optionally an icon URL for this license.
- Using the Dataverse API to register the new license as one of the options available in your installation.
- Using the API to make sure the license is active and deciding whether the license should also be the default.
- Once the license is registered with Dataverse, making an SQL update to change datasets/versions using that license to reference it instead of having their own copy of those custom terms.

The benefits of this approach are:

- usability: the license can be selected for new datasets without allowing custom terms and without users having to cut/paste terms or collection administrators having to configure templates with those terms
- efficiency: custom terms are stored per dataset whereas licenses are registered once and all uses of it refer to the same object and external URL
- security: with the license terms maintained external to Dataverse, users cannot edit specific terms and curators do not need to check for edits

Once a standardized version of you Custom Terms are registered as a license, an SQL update like the following can be used to have datasets use it:

```
UPDATE termsofuseandaccess
    SET license_id = (SELECT license.id FROM license WHERE license.name = '<Your License␣
↪Name>'), termsofuse=null, confidentialitydeclaration=null, t.specialpermissions=null,␣
↪t.restrictions=null, citationrequirements=null, depositorrequirements=null,␣
↪conditions=null, disclaimer=null
    WHERE termsofuseandaccess.termsofuse LIKE '%<Unique phrase in your Terms of Use>%';
```

6.14.3 Optional Components

Standalone "Zipper" Service Tool

As of Dataverse Software 5.0 we offer an **experimental** optimization for the multi-file, download-as-zip functionality. If this option (`:CustomZipDownloadServiceUrl`) is enabled, instead of enforcing the size limit on multi-file zipped downloads (as normally specified by the option `:ZipDownloadLimit`), we attempt to serve all the files that the user requested (that they are authorized to download), but the request is redirected to a standalone zipper service running as a cgi-bin executable under Apache. This moves these potentially long-running jobs completely outside the Application Server (Payara), and prevents worker threads from becoming locked serving them. Since zipping is also a CPU-intensive task, it is possible to have this service running on a different host system, freeing the cycles on the main Application Server. (The system running the service needs to have access to the database as well as to the storage filesystem, and/or S3 bucket).

Please consult the README at scripts/zipdownload in the Dataverse Software 5.0+ source tree for more information.

To install:

1. Follow the instructions in the file above to build `zipdownloader-0.0.1.jar`. Please note that the package name and the version were changed as of the release 5.10, as part of an overall cleanup and reorganization of the project tree. In the releases 5.0-5.9 it existed under the name `ZipDownloadService-v1.0.0`. (A pre-built jar file was distributed under that name as part of the 5.0 release on GitHub. Aside from the name change, there have been no changes in the functionality of the tool).
2. Copy it, together with the shell script `cgi-bin/zipdownload` to the `cgi-bin` directory of the chosen Apache server (`/var/www/cgi-bin` standard).
3. Make sure the shell script (`zipdownload`) is executable, and edit it to configure the database access credentials. Do note that the executable does not need access to the entire Dataverse installation database. A security-conscious admin can create a dedicated database user with access to just one table: `CUSTOMZIPSERVICEREQUEST`.

You may need to make extra Apache configuration changes to make sure `/cgi-bin/zipdownload` is accessible from the outside. For example, if this is the same Apache that's in front of your Dataverse installation Payara instance, you will need to add another pass through statement to your configuration:

```
ProxyPassMatch ^/cgi-bin/zipdownload !
```

Test this by accessing it directly at `<SERVER URL>/cgi-bin/download`. You should get a `404 No such download job!`. If instead you are getting an "internal server error", this may be an SELinux issue; try `setenforce Permissive`. If you are getting a generic Dataverse collection "not found" page, review the `ProxyPassMatch` rule you have added.

To activate in your Dataverse installation:

```
curl -X PUT -d '/cgi-bin/zipdownload' http://localhost:8080/api/admin/settings/
↪:CustomZipDownloadServiceUrl
```

External Metadata Exporters

Dataverse 5.14+ supports the configuration of external metadata exporters (just "external exporters" or "exporters" for short) as a way to add additional metadata export formats or replace built-in formats. For a list of built-in formats, see *Supported Metadata Export Formats* in the User Guide.

This should be considered an **experimental** capability in that the mechanism is expected to evolve and using it may require additional effort when upgrading to new Dataverse versions.

Enabling External Exporters

Use the *dataverse.spi.exporters.directory* configuration option to specify a directory from which external exporters (JAR files) should be loaded.

Inventory of External Exporters

For a list of external exporters, see the README at https://github.com/gdcc/dataverse-exporters. For example:

- RO-Crate

Developing New Exporters

See *Metadata Export Formats* for details about how to develop new exporters.

CHAPTER

SEVEN

CONTRIBUTOR GUIDE

Thank you for your interest in contributing to Dataverse! We are open to contributions from everyone. We welcome contributions of ideas, bug reports, documentation, code, and more!

Contents:

- *Ideas and Feature Requests*
- *Bug Reports*
- *Documentation*
- *Translations*
- *Code*
- *Usability Testing*
- *Answering Questions*
- *Sample Data*
- *Issue Triage*
- *Giving Talks*
- *Working Groups*
- *GDCC*
- *Artwork*
- *Other Contributions*

7.1 Ideas and Feature Requests

We would love to hear your ideas!

1. Please check if your idea or feature request is already captured in our issue tracker or roadmap.
2. Bring your idea to the community by posting on our Google Group or chat.dataverse.org.
3. To discuss privately, email us at support@dataverse.org.

7.2 Bug Reports

Before submitting an issue, please search for existing issues in our issue tracker. If there is an existing open issue that matches the issue you want to report, please add a comment to it or give it a (thumbs up).

If there is no pre-existing issue or it has been closed, please open a new issue (unless it is a security issue which should be reported privately to security@dataverse.org as discussed under *Reporting Security Issues* in the Installation Guide).

If you do not receive a reply to your new issue or comment in a timely manner, please ping us at chat.dataverse.org.

7.3 Documentation

Documentation is such a large topic (and important!) that we have a dedicate section on it:

7.3.1 Writing Documentation

Thank you for your interest in contributing documentation to Dataverse! Good documentation is absolutely critical to the success of software.

Contents:

- *Overview*
- *Quick Fix*
- *Building the Guides with Sphinx*
  - *Docker*
    - *Docker with a Makefile*
  - *Sphinx Installed Locally*
- *Editing, Building, and Previewing the Guides*
- *Making a Pull Request*
- *Writing Guidelines*
  - *Writing Style Guidelines*
  - *Table of Contents*
  - *Links*
    - *Custom Titles*
  - *Images*
  - *Cross References*
- *PDF Version of the Guides*
- *Hosting Your Own Version of the Guides*
  - *Having Google Index the Latest Version*

Overview

The Dataverse guides are written using Sphinx.

The source files are stored under doc/sphinx-guides in the main "dataverse" repo on GitHub.

Historically, guides have been written in the default Sphinx format, reStructuredText (.rst), but newer guides such as the *Contributor Guide* are written in Markdown (.md).

Below we'll present a technique for making quick edits to the guides using GitHub's web editor ("quick fix"). We'll also describe how to install Sphinx locally for more significant edits.

Finally, we'll provide some guidelines on writing content.

We could use some help on writing this very page and helping newcomers get started. Please don't be shy about suggesting improvements! You can open an issue at https://github.com/IQSS/dataverse/issues, post to https://chat.dataverse.org, write to the mailing list, or suggest a change with a pull request.

Quick Fix

If you find a typo or a small error in the documentation you can fix it using GitHub's online web editor. Generally speaking, we will be following GitHub's guidance on editing files in another user's repository.

- Navigate to https://github.com/IQSS/dataverse/tree/develop/doc/sphinx-guides/source where you will see folders for each of the guides: admin, api, container, etc.
- Find the file you want to edit under one of the folders above.
- Click the pencil icon in the upper-right corner. If this is your first contribution to Dataverse, the hover text over the pencil icon will say "Fork this project and edit this file".
- Make changes to the file and preview them.
- In the **Commit changes** box, enter a description of the changes you have made and click **Propose file change**.
- Under the **Write** tab, delete the long welcome message and write a few words about what you fixed.
- Click **Create Pull Request**.

That's it! Thank you for your contribution! Your pull request will be added manually to the main Dataverse project board at https://github.com/orgs/IQSS/projects/34 and will go through code review and QA before it is merged into the "develop" branch. Along the way, developers might suggest changes or make them on your behalf. Once your pull request has been merged you will be listed as a contributor at https://github.com/IQSS/dataverse/graphs/contributors!

Please see https://github.com/IQSS/dataverse/pull/5857 for an example of a quick fix that was merged (the "Files changed" tab shows how a typo was fixed).

Preview your documentation changes which will be built automatically as part of your pull request in Github. It will show up as a check entitled: `docs/readthedocs.org:dataverse-guide - Read the Docs build succeeded!`. For example, this PR built to https://dataverse-guide--9249.org.readthedocs.build/en/9249/.

If you would like to read more about the Dataverse's use of GitHub, please see the *Version Control* section of the Developer Guide. For bug fixes and features we request that you create an issue before making a pull request but this is not at all necessary for quick fixes to the documentation.

Building the Guides with Sphinx

While the "quick fix" technique shown above should work fine for minor changes, in many cases, you're going to want to preview changes locally before committing them.

Before we worry about pushing changes to the code, let's make sure we can build the guides.

Go to https://github.com/IQSS/dataverse and click "Code" and then follow the instructions to clone the code locally.

Docker

Install Docker Desktop.

From a terminal, switch to the "dataverse" directory you just cloned. This is the root of the git repo.

```
cd dataverse
```

Then try running this command:

```
docker run -it --rm -v $(pwd):/docs sphinxdoc/sphinx:7.4.0 bash -c "cd doc/sphinx-guides
&& pip3 install -r requirements.txt && make html"
```

If all goes well, you should be able to open `doc/sphinx-guides/build/html/index.html` to see the guides you just built.

Docker with a Makefile

Once you've confirmed you have Docker working, if you have make installed, you can try the following commands:

```
make docs-html
```

```
make docs-pdf
```

```
make docs-epub
```

```
make docs-all
```

Sphinx Installed Locally

First, run `python --version` or `python3 --version` to determine the version of Python you have. If you don't have Python 3.10 or higher, you must upgrade.

Next, change to the `doc/sphinx-guides` directory.

```
cd doc/sphinx-guides
```

Create a Python virtual environment, activate it, then install dependencies:

```
python3 -m venv venv
```

```
source venv/bin/activate
```

```
pip install -r requirements.txt
```

Next, install GraphViz because building the guides requires having the `dot` executable from GraphViz either on the path or passed as an argument.

On a Mac we recommend installing GraphViz through Homebrew. Once you have Homebrew installed and configured to work with your shell, you can type `brew install graphviz`.

Finally, you can try building the guides with the following command.

```
make html
```

If all goes well, you should be able to open `doc/sphinx-guides/build/html/index.html` to see the guides you just built.

Editing, Building, and Previewing the Guides

To edit the existing documentation:

- Create a branch (see *How to Make a Pull Request*).
- In `doc/sphinx-guides/source` you will find the .rst or .md files that correspond to https://guides.dataverse.org.
- Using your preferred text editor, open and edit the necessary files, or create new ones.

Once you are done, you can preview the changes by building the guides using one of the options above.

After Sphinx is done processing the files you should notice that the `html` folder in `doc/sphinx-guides/build` directory has been updated. You can click on the files in the `html` folder to preview the changes.

Making a Pull Request

Now you can make a commit with the changes to your own fork in GitHub and submit a pull request. See *How to Make a Pull Request*.

Writing Guidelines

Writing Style Guidelines

Please observe the following when writing documentation:

- Use American English spelling.
- Use examples when possible.
- Break up longer paragraphs.
- Use Title Case in Headings.
- Use “double quotes” instead of ‘single quotes’.
- Favor “and” (data and code) over slashes (data/code).

Table of Contents

Every non-index page should use the following code to display a table of contents of internal sub-headings. This code should be placed below any introductory text and directly above the first subheading, much like a Wikipedia page.

If the page is written in reStructuredText (.rst), use this form:

```
.. contents:: |toctitle|
   :local:
```

If the page is written in Markdown (.md), use this form:

```
```{contents} Contents:
:local:
:depth: 3
```
```

Links

Getting links right can be tricky.

Custom Titles

In .rst files you can use a custom title when linking to a document like this:

```
:doc:`Custom title </api/intro>`
```

See also https://docs.readthedocs.io/en/stable/guides/cross-referencing-with-sphinx.html#the-doc-role

In .md files, the same pattern can be used. Here's an example of using a custom title with a ref:

```
{ref}`Log in <account-log-in-options>`
```

See also https://myst-parser.readthedocs.io/en/v0.16.1/syntax/syntax.html#targets-and-cross-referencing

Images

A good documentation is just like a website enhanced and upgraded by adding high quality and self-explanatory images. Often images depict a lot of written text in a simple manner. Within our Sphinx docs, you can add them in two ways: a) add a PNG image directly and include or b) use inline description languages like GraphViz (current only option).

While PNGs in the git repo can be linked directly via URL, Sphinx-generated images do not need a manual step and might provide higher visual quality. Especially in terms of quality of content, generated images can be extendend and improved by a textbased and reviewable commit, without needing raw data or source files and no diff around.

Cross References

When adding ReStructured Text (.rst) cross references, use the hyphen character (`-`) as the word separator for the cross reference label. For example, `my-reference-label` would be the preferred label for a cross reference as opposed to, for example, `my_reference_label`.

PDF Version of the Guides

The HTML version of the guides is the official one. Any other formats are maintained on a best effort basis.

If you would like to build a PDF version of the guides and have Docker installed, please try the command below from the root of the git repo:

```
docker run -it --rm -v $(pwd):/docs sphinxdoc/sphinx-latexpdf:7.4.0 bash -c "cd doc/
sphinx-guides && pip3 install -r requirements.txt && make latexpdf LATEXMKOPTS=\
"-interaction=nonstopmode\"; cd ../.. && ls -1 doc/sphinx-guides/build/latex/Dataverse.
pdf"
```

A few notes about the command above:

- Hopefully the PDF was created at `doc/sphinx-guides/build/latex/Dataverse.pdf`.
- For now, we are using "nonstopmode" but this masks some errors.
- See requirements.txt for a note regarding the version of Sphinx we are using.

Also, as of this writing we have enabled PDF builds from the "develop" branch. You download the PDF from http://preview.guides.gdcc.io/_/downloads/en/develop/pdf/

If you would like to help improve the PDF version of the guides, please get in touch! Please see *Getting Help* for ways to contact the developer community.

Hosting Your Own Version of the Guides

Some installations of Dataverse maintain their own versions of the guides and use settings like *:NavbarGuidesUrl* or *:GuidesBaseUrl* to point their users to them.

Having Google Index the Latest Version

As each version of the Dataverse software is released, there is an updated version of the guides released with it. Google and other search engines index all versions, which may confuse users who discover your guides in the search results as to which version they should be looking at. When learning about your installation from the search results, it is best to be viewing the *latest* version.

In order to make it clear to the crawlers that we only want the latest version discoverable in their search results, we suggest adding this to your `robots.txt` file:

```
User-agent: *
Allow: /en/latest/
Disallow: /en/
```

7.4 Translations

If you speak multiple languages, you are very welcome to help us translate Dataverse! Please see *How to Help Translate the Dataverse Software Into Your Language* for details.

7.5 Code

Dataverse is open source and we love code contributions. Developers are not limited to the main Dataverse code in this git repo. We have projects in C, C++, Go, Java, Javascript, Julia, PHP, Python, R, Ruby, Rust, TypeScript and more. To get started, please see the following pages:

7.5.1 Contributing Code

We love code contributions! There are lots of ways you can help.

Contents:

- *Finding an Issue to Work On*
  - *Many Codebases, Many Languages*
  - *Picking a Good First Issue*
- *Making a Pull Request*
- *Reviewing Code*
- *Reproducing Bugs*
- *Getting Help*

Finding an Issue to Work On

New contributors often wonder what issues they should work on first.

Many Codebases, Many Languages

The primary codebase and issue tracker for Dataverse is https://github.com/IQSS/dataverse. It's mostly backend code written in Java. However, there are many other codebases you can work on in a variety of languages. Here are a few that are especially active:

- https://github.com/IQSS/dataverse (Java)
- https://github.com/IQSS/dataverse-frontend (React)
- https://github.com/IQSS/dataverse-client-javascript (TypeScript)
- https://github.com/gdcc/dataverse-previewers (Javascript)
- https://github.com/gdcc/pyDataverse (Python)
- https://github.com/gdcc/rust-dataverse (Rust)
- https://github.com/gdcc/dataverse-ansible (Ansible)
- https://github.com/gdcc/dv-metrics (Javascript)

If nothing above sparks joy, you can find more projects to work on under *Client Libraries*, *Building External Tools*, *Related Projects*, and *Apps*.

Picking a Good First Issue

Once you've decided which codebase suits you, you should try to identify an issue to work on. Some codebases use a label like "good first issue" to suggest issues for newcomers.

For the main codebase, please see *Finding GitHub Issues to Work On* which includes information on labels like "good first issue".

Other codebases may use different labels. Check the README or other documentation for that codebase.

If there is a linked pull request that is trying to close the issue, you should probably find another issue.

If you are having trouble finding an issue or have any questions at all, please do not hesitate to reach out. See *Getting Help*.

Making a Pull Request

For the main codebase, please see *How to Make a Pull Request*.

For other codebases, consult the README.

Reviewing Code

Reviewing code is a great way to learn about a codebase. For any codebase you can browse open pull requests, of course, but for the primary codebases, you can take a look at the "Ready for Review" and "In Review" columns at https://github.com/orgs/IQSS/projects/34.

You are welcome to review code informally or to leave an actual review. We're interested in what you think.

Reproducing Bugs

At times, bugs are reported that we haven't had time to confirm. You can help out by reproducing bugs and commenting on the issue the results you find.

Getting Help

If you have any questions at all, please do not hesitate to reach out. See *Getting Help*.

7.6 Usability Testing

Please email us at support@dataverse.org or fill in our feedback form if you are interested in participating in usability testing.

7.7 Answering Questions

People come along with questions on the mailing list and in chat all the time. You are very welcome to help out by answering these questions to the best of your ability.

7.8 Sample Data

Consider contributing to https://github.com/IQSS/dataverse-sample-data, a git repo of realistic-looking data that is used for testing.

7.9 Issue Triage

New issues come in all the time, especially for the main issue tracker at https://github.com/IQSS/dataverse/issues.

You can help by leaving comments. You can mention related issues or answer questions.

If you are interested in adding issue labels or related curation, please get in touch!

7.10 Giving Talks

If you give a recorded talk about Dataverse, we are happy to add it to DataverseTV. You can just leave a comment on the spreadsheet or make some noise in chat.

For non-recorded talks, we are happy to upload your slides to https://dataverse.org/presentations. Please email support@dataverse.org.

7.11 Working Groups

Most working groups are wide open to participation. For the current list of groups, please see https://www.gdcc.io/working-groups.html.

You're welcome to start your own working group, of course. We can help you get the word out.

7.12 GDCC

The popularity of the Dataverse software has resulted in a continuously growing community with different needs and requirements. The Global Dataverse Community Consortium (GDCC) helps coordinate community efforts and sustain the software and community in the long term. Please consider contributing to GDCC by joining as an institutional member (https://www.gdcc.io/members.html).

7.13 Artwork

Surely we can put artistic talent to use. A contributor drew cartoon animals chatting about Dataverse, for example.

As annnounced, we have a sticker template you can use.

See Illustrations from The Turing Way for how that community has created artwork. Perhaps we can create a similar collection.

7.14 Other Contributions

We consulted https://opensource.guide/how-to-contribute/ but no list is comprehensive.

What else should we list here? How to YOU want to contribute to Dataverse?

CHAPTER

EIGHT

DEVELOPER GUIDE

Contents:

8.1 Introduction

Welcome! The Dataverse Project is an open source project that loves contributors!

Contents:

- *Intended Audience*
- *Getting Help*
- *Core Technologies*
- *Roadmap*
- *Kanban Board*
- *Issue Tracker*
- *Related Guides*
- *Related Projects*

8.1.1 Intended Audience

This guide is intended primarily for developers who want to work on the main Dataverse codebase at https://github.com/IQSS/dataverse but see the *Contributing Code* section of the Contributor Guide for other code you can work on!

To get started, you'll want to set up your *Development Environment* and make sure you understand the branching strategy described in the *Version Control* section and how to make a pull request. *Testing* is expected. Opinions about *Coding Style* are welcome!

8.1.2 Getting Help

If you have any questions at all, please reach out to other developers via https://chat.dataverse.org, the dataverse-dev mailing list, the dataverse-community mailing list, or community calls.

8.1.3 Core Technologies

Dataverse is a Jakarta EE application that is compiled into a WAR file and deployed to an application server (app server) which is configured to work with a relational database (PostgreSQL) and a search engine (Solr).

We make use of a variety of Jakarta EE technologies such as JPA, JAX-RS, JMS, and JSF. In addition, we use parts of Eclipse MicroProfile such as MicroProfile Config.

The frontend is built using PrimeFaces and Bootstrap. A new frontend is being built using React at https://github.com/IQSS/dataverse-frontend

8.1.4 Roadmap

For the roadmap, please see https://www.iq.harvard.edu/roadmap-dataverse-project

8.1.5 Kanban Board

You can get a sense of what's currently in flight (in dev, in QA, etc.) by looking at https://github.com/orgs/IQSS/projects/34

8.1.6 Issue Tracker

The main issue tracker is https://github.com/IQSS/dataverse/issues but note that individual projects have their own issue trackers.

8.1.7 Related Guides

If you are wondering about how to contribute generally, please see the *Contributor Guide*.

If you are a developer who wants to make use of the Dataverse APIs, please see the *API Guide*.

If you have frontend UI questions, please see the *Style Guide*. For the new frontend, see https://github.com/IQSS/dataverse-frontend

If you are a Docker enthusiasts, please check out the *Container Guide*.

8.1.8 Related Projects

Note: this list is somewhat old. Please see also the *Contributing Code* section of the Contributor Guide.

As a developer, you also may be interested in these projects related to Dataverse:

- External Tools - add additional features to the Dataverse Software without modifying the core: *Building External Tools*
- Dataverse Software API client libraries - use Dataverse Software APIs from various languages: *Client Libraries*

- DVUploader - a stand-alone command-line Java application that uses the Dataverse Software API to support upload of files from local disk to a Dataset: https://github.com/IQSS/dataverse-uploader
- dataverse-sample-data - populate your Dataverse installation with sample data: https://github.com/IQSS/dataverse-sample-data
- dataverse-metrics - aggregate and visualize metrics for Dataverse installations around the world: https://github.com/IQSS/dataverse-metrics
- Configuration management scripts - Ansible, Puppet, etc.: See *Advanced Installation* section in the Installation Guide.
- *Universal Numerical Fingerprint (UNF)* (Java) - a Universal Numerical Fingerprint: https://github.com/IQSS/UNF
- DataTags (Java and Scala) - tag datasets with privacy levels: https://github.com/IQSS/DataTags
- Matrix - a visualization showing the connectedness and collaboration between authors and their affiliations.
- Third party apps - make use of Dataverse installation APIs: *Apps*
- chat.dataverse.org - chat interface for Dataverse Project users and developers: https://github.com/IQSS/chat.dataverse.org
- [Your project here] :)

8.2 Development Environment

These instructions are oriented around Docker but the "classic" instructions we used for Dataverse 4 and 5 are still available at *Classic Dev Environment*.

Contents:

- *Quickstart*
- *Detailed Steps*
  - *Install Java*
  - *Install Maven*
  - *Install and Start Docker*
  - *Git Clone Repo*
  - *Build and Run*
  - *Verify*
- *Next Steps*
- *Getting Help*

8.2.1 Quickstart

First, install Java 21, Maven, and Docker.

After cloning the dataverse repo, run this:

```
mvn -Pct clean package docker:run
```

(Note that if you are Windows, you must run the command above in WSL rather than cmd.exe. See *Windows Development*.)

After some time you should be able to log in:

- url: http://localhost:8080
- username: dataverseAdmin
- password: admin1

8.2.2 Detailed Steps

Install Java

The recommended version is Java 21 because it's the version we test with. See https://github.com/IQSS/dataverse/pull/12043.

On Mac and Windows, we suggest using SDKMAN to install Temurin (Eclipe's name for its OpenJDK distribution). Type `sdk install java 21` and then hit the "tab" key until you get to a version that ends with `-tem` and then hit enter. If you don't set this version as the default, you will need to type `sdk use java 21` and then hit the tab to autocomplete the version.

Alternatively you can download Temurin from https://adoptium.net (formerly AdoptOpenJDK).

On Linux, you are welcome to use the OpenJDK available from package managers.

Install Maven

If you are using SKDMAN, run this command:

```
sdk install maven
```

Otherwise, follow instructions at https://maven.apache.org.

Install and Start Docker

Follow instructions at https://www.docker.com

Be sure to start Docker.

Git Clone Repo

Fork https://github.com/IQSS/dataverse and then clone your fork like this:

```
git clone git@github.com:[YOUR GITHUB USERNAME]/dataverse.git
```

Build and Run

Change into the `dataverse` directory you just cloned and run the following command:

```
mvn -Pct clean package docker:run
```

Verify

After some time you should be able to log in:

- url: http://localhost:8080
- username: dataverseAdmin
- password: admin1

8.2.3 Next Steps

See the *Development Usage* section of the Container Guide for tips on fast redeployment, viewing logs, and more.

8.2.4 Getting Help

Please feel free to reach out at https://chat.dataverse.org or https://groups.google.com/g/dataverse-dev if you have any difficulty setting up a dev environment!

8.3 Windows Development

Contents:

- *Running Dataverse in Windows WSL*
  - *Install Docker Desktop*
  - *Install WSL*
  - *Prepare WSL*
  - *Install Dataverse*
  - *IDEs for Dataverse in Windows*
  - *pgAdmin in Windows for Dataverse*

8.3.1 Running Dataverse in Windows WSL

The simplest method to run Dataverse in Windows 10 and 11 is using Docker and Windows Subsystem for Linux (WSL) - specifically WSL 2. Once Docker and WSL are installed, you can follow the *quickstart instructions*.

Please note: these instructions have not been extensively tested. They have been found to work with the Ubuntu-24.04 distribution for WSL. If you find any problems, please open an issue at https://github.com/IQSS/dataverse/issues and/or submit a PR to update this guide.

Install Docker Desktop

Follow the directions at https://www.docker.com to install Docker Desktop on Windows. If prompted, turn on WSL 2 during installation.

Settings you may need in Docker Desktop:

- **General/Expose daemon on tcp://localhost:2375 without TLS**: true
- **General/Use the WSL 2 based engine**: true
- **General/Add the *.docker.internal names to the host's /etc/hosts file (Requires password)**: true
- **Resources/WSL Integration/Enable integration with my default WSL distro**: true
- **Resources/WSL Integration/Enable integration with additional distros**: select any you run Dataverse in

Install WSL

If you install Docker Desktop, you should already have WSL installed. If not, or if you wish to add an additional Linux distribution, open PowerShell.

If WSL itself is not installed run:

```
wsl --install
```

For use with Docker, you should use WSL v2 - run:

```
wsl  --set-default-version 2
```

Install a specific Linux distribution. To see the list of possible distributions:

```
wsl --list --online
```

Choose the distribution you would like. Then run the following command. These instructions were tested with `Ubuntu 24.04 LTS`.

```
wsl --install -d <Distribution Name>
```

You will be asked to create an initial Linux user.

> **Note**
>
> Using wsl –set-version to upgrade an existing distribution from WSL 1 to WSL 2 may not work - installing a new distribution using WSL 2 is recommended.

Prepare WSL

Once that you have WSL installed, You will need Java and MVN working inside WSL, how you go about this will depend on the Linux distribution you installed in WSL.

Here is an example using SDKMAN, which is not required, but it is recommended for managing Java and other SDKs.

```
sudo apt update
sudo apt install zip
```

```
sudo apt update
sudo apt install unzip
```

```
curl -s "https://get.sdkman.io" | bash
source "$HOME/.sdkman/bin/sdkman-init.sh"
```

```
sdk install java 21
```

```
sdk install maven
```

Install Dataverse

Open a Linux terminal (e.g. use Windows Terminal and open a tab for the Linux distribution you selected). Then install Dataverse in WSL following the *quickstart instructions*. You should then have a working Dataverse instance.

We strongly recommend that you clone the Dataverse repository from WSL, not from Windows. This will ensure that builds are much faster.

IDEs for Dataverse in Windows

You can use your favorite editor or IDE to edit Dataverse project files. Files in WSL are accessible from Windows for editing using the path `\\wsl.localhost`. Your Linux distribution files should also be visible in File Explorer under the This PC/Linux entry.

Note

FYI: For the best performance, it is recommended, with WSL 2, to store Dataverse files in the WSL/Linux file system and to access them from there with your Windows-based IDE (versus storing Dataverse files in your Windows file system and trying to run maven and build from Linux - access to /mnt/c files using WSL 2 is slow).

pgAdmin in Windows for Dataverse

You can access the Dataverse database from Windows.

Install pgAdmin from https://www.pgadmin.org/download/pgadmin-4-windows/

In pgAdmin, register a server using `127.0.0.1` with port `5432`. For the database name, username, and password, see *Database Name and Credentials*. Now you will be able to access, monitor, and update the Dataverse database.

8.4 Tips

If you just followed the steps in *Classic Dev Environment* for the first time, you will need to get set up to deploy code to your app server. Below you'll find other tips as well.

Contents:

- *Iterating on Code and Redeploying*
  - *Undeploy the war File from the Dataverse Software Installation Script*
  - *Add Payara as a Server in Netbeans*
  - *Ensure that the Dataverse Software Will Be Deployed to Payara*
  - *Make a Small Change to the Code*
  - *Confirm the Change Was Deployed*
- *Netbeans Connector Chrome Extension*
- *Thumbnails*
- *Database Schema Exploration*
  - *Database Name and Credentials*
  - *pgAdmin*
  - *SchemaSpy*
- *Deploying With* `asadmin`
- *Running the Dataverse Software Installation Script in Non-Interactive Mode*
- *Preventing Payara from Phoning Home*
- *Solr*
  - *Updating the Solr Schema (Developers)*
- *Git*
  - *Set Up SSH Keys*
  - *Git on Mac*
  - *Automation of Custom Build Number on Webpage*
- *Sample Data*
- *Switching from Glassfish to Payara*
- *UI Pages Development*
  - *Avoiding Inefficiencies in JSF Render Logic*
  - *JSF1103 Errors*

8.4.1 Iterating on Code and Redeploying

When you followed the steps in the *Classic Dev Environment* section, the war file was deployed to Payara by the Dataverse Software installation script. That's fine but once you're ready to make a change to the code you will need to get comfortable with undeploying and redeploying code (a war file) to Payara.

It's certainly possible to manage deployment and undeployment of the war file via the command line using the `asadmin` command that ships with Payara (that's what the Dataverse Software installation script uses and the steps are documented below), but we recommend getting set up with an IDE such as Netbeans to manage deployment for you.

Undeploy the war File from the Dataverse Software Installation Script

Because the initial deployment of the war file was done outside of Netbeans by the Dataverse Software installation script, it's a good idea to undeploy that war file to give Netbeans a clean slate to work with.

Assuming you installed Payara in `/usr/local/payara7`, run the following `asadmin` command to see the version of the Dataverse Software that the Dataverse Software installation script deployed:

```
/usr/local/payara7/bin/asadmin list-applications
```

You will probably see something like `dataverse-5.0 <ejb, web>` as the output. To undeploy, use whichever version you see like this:

```
/usr/local/payara7/bin/asadmin undeploy dataverse-5.0
```

Now that Payara doesn't have anything deployed, we can proceed with getting Netbeans set up to deploy the code.

Add Payara as a Server in Netbeans

Launch Netbeans and click "Tools" and then "Servers". Click "Add Server" and select "Payara Server" and set the installation location to `/usr/local/payara7`. The defaults are fine so you can click "Next" and "Finish".

Please note that if you are on a Mac, Netbeans may be unable to start Payara due to proxy settings in Netbeans. Go to the "General" tab in Netbeans preferences and click "Test connection" to see if you are affected. If you get a green checkmark, you're all set. If you get a red exclamation mark, change "Proxy Settings" to "No Proxy" and retest. A more complicated answer having to do with changing network settings is available at https://discussions.apple.com/thread/7680039?answerId=30715103022#30715103022 and the bug is also described at https://netbeans.org/bugzilla/show_bug.cgi?id=268076

At this point you can manage Payara using Netbeans. Click "Window" and then "Services". Expand "Servers" and right-click Payara to stop and then start it so that it appears in the Output window. Note that you can expand "Payara" and "Applications" to see if any applications are deployed.

Ensure that the Dataverse Software Will Be Deployed to Payara

Click "Window" and then "Projects". Click "File" and then "Project Properties (dataverse)". Click "Run" and change "Server" from "No Server Selected" to your installation of Payara. Click OK.

Make a Small Change to the Code

Let's make a tiny change to the code, compile the war file, deploy it, and verify that that we can see the change.

One of the smallest changes we can make is adjusting the build number that appears in the lower right of every page.

From the root of the git repo, run the following command to set the build number to the word "hello" (or whatever you want):

```
scripts/installer/custom-build-number hello
```

This should update or place a file at `src/main/java/BuildNumber.properties`.

(See also *Automation of Custom Build Number on Webpage* for other ways of changing the build number.)

Then, from Netbeans, click "Run" and then "Clean and Build Project (dataverse)". After this completes successfully, click "Run" and then "Run Project (dataverse)"

Confirm the Change Was Deployed

After deployment, check the build number in the lower right to make sure it has been customized. You can also check the build number by running the following command:

```
curl http://localhost:8080/api/info/version
```

If you can see the change, great! Please go fix a bug or work on a feature! :)

Actually, before you start changing any code, you should create a branch as explained in the *Version Control* section.

While it's fresh in your mind, if you have any suggestions on how to make the setup of a development environment easier, please get in touch!

8.4.2 Netbeans Connector Chrome Extension

For faster iteration while working on JSF pages, it is highly recommended that you install the Netbeans Connector Chrome Extension listed in the *Tools* section. When you save XHTML or CSS files, you will see the changes immediately. Hipsters call this "hot reloading". :)

8.4.3 Thumbnails

In order for thumnails to be generated for PDFs, you need to install ImageMagick and configure Dataverse to use the `convert` binary.

Assuming you're using Homebrew:

```
brew install imagemagick
```

Then configure the JVM option mentioned in *ImageMagick* to the path to `convert` which for Homebrew is usually `/usr/local/bin/convert`.

8.4.4 Database Schema Exploration

With over 100 tables, the Dataverse PostgreSQL database can be somewhat daunting for newcomers. Here are some tips for coming up to speed. (See also the *SQL Upgrade Scripts* section.)

Database Name and Credentials

The default database name and credentials depends on how you set up your dev environment.

MPCONFIG Key	Docker	Classic
dataverse.db.name	`dataverse`	`dvndb`
dataverse.db.user	`dataverse`	`dvnapp`
dataverse.db.password	`secret`	`secret`

Here's an example of using these credentials from within the PostgreSQL container (see *Container Guide*):

```
pdurbin@beamish dataverse % docker exec -it dev_postgres bash
root@postgres:/# export PGPASSWORD=secret
root@postgres:/# psql -h localhost -U dataverse dataverse
psql (16.3 (Debian 16.3-1.pgdg120+1))
Type "help" for help.

dataverse=# select id,alias from dataverse limit 1;
 id | alias
----+-------
  1 | root
(1 row)
```

See also *Database Persistence* in the Installation Guide.

pgAdmin

If you followed the *Classic Dev Environment* section, we had you install pgAdmin, which can help you explore the tables and execute SQL commands. It's also listed in the *Tools* section.

SchemaSpy

SchemaSpy is a tool that creates a website of entity-relationship diagrams based on your database.

As part of our release process (*Update SchemaSpy*), we run SchemaSpy and publish the output at https://guides.dataverse.org/en/latest/schemaspy/index.html and (for example) https://guides.dataverse.org/en/6.6/schemaspy/index.html

To run SchemaSpy locally, you can try something like this (after downloading the jars from https://github.com/schemaspy/schemaspy/releases and https://jdbc.postgresql.org/download/):

```
java -jar /tmp/schemaspy-6.2.4.jar -t pgsql -host localhost -db dvndb -u postgres -p
secret -s public -dp /tmp/postgresql-42.7.5.jar -o /tmp/latest
```

See also *Database Name and Credentials*.

8.4.5 Deploying With `asadmin`

Sometimes you want to deploy code without using Netbeans or from the command line on a server you have ssh'ed into.

For the `asadmin` commands below, we assume you have already changed directories to `/usr/local/payara7/glassfish/bin` or wherever you have installed Payara.

There are four steps to this process:

1. Build the war file: `mvn package`
2. Check which version of the Dataverse Software is deployed: `./asadmin list-applications`
3. Undeploy the Dataverse Software (if necessary): `./asadmin undeploy dataverse-VERSION`
4. Copy the war file to the server (if necessary)
5. Deploy the new code: `./asadmin deploy /path/to/dataverse-VERSION.war`

8.4.6 Running the Dataverse Software Installation Script in Non-Interactive Mode

Rather than running the installer in "interactive" mode, it's possible to put the values in a file. See "non-interactive mode" in the *Installation* section of the Installation Guide.

8.4.7 Preventing Payara from Phoning Home

By default, Glassfish reports analytics information. The administration guide suggests this can be disabled with `./asadmin create-jvm-options -Dcom.sun.enterprise.tools.admingui.NO_NETWORK=true`, should this be found to be undesirable for development purposes. It is unknown if Payara phones home or not.

8.4.8 Solr

Once some Dataverse collections, datasets, and files have been created and indexed, you can experiment with searches directly from Solr at http://localhost:8983/solr/#/collection1/query and look at the JSON output of searches, such as this wildcard search: http://localhost:8983/solr/collection1/select?q=*%3A*&wt=json&indent=true . You can also get JSON output of static fields Solr knows about: http://localhost:8983/solr/collection1/schema/fields

You can simply double-click "start.jar" rather than running `java -jar start.jar` from the command line. Figuring out how to stop Solr after double-clicking it is an exercise for the reader.

Updating the Solr Schema (Developers)

Both developers and sysadmins need to update the Solr schema from time to time. One difference is that developers will be committing changes to `conf/solr/schema.xml` in git. To prevent cross-platform differences in the git history, when running the `update-fields.sh` script, we ask all developers to run the script from within Docker. (See *Config Baker Image* for more on the image we'll use below.)

```
curl http://localhost:8080/api/admin/index/solr/schema | docker run -i --rm -v ./docker-
↪dev-volumes/solr/data:/var/solr gdcc/configbaker:unstable update-fields.sh /var/solr/
↪data/collection1/conf/schema.xml

cp docker-dev-volumes/solr/data/data/collection1/conf/schema.xml conf/solr/schema.xml
```

At this point you can do a `git diff` and see if your changes make sense before committing.

Sysadmins are welcome to run `update-fields.sh` however they like. See *Updating the Solr Schema* in the Admin Guide and *Additional Metadata Blocks* in the Container Guide for details.

8.4.9 Git

Set Up SSH Keys

You can use git with passwords over HTTPS, but it's much nicer to set up SSH keys. https://github.com/settings/ssh is the place to manage the ssh keys GitHub knows about for you. That page also links to a nice howto: https://help.github.com/articles/generating-ssh-keys

From the terminal, `ssh-keygen` will create new ssh keys for you:

- private key: `~/.ssh/id_rsa` - It is very important to protect your private key. If someone else acquires it, they can access private repositories on GitHub and make commits as you! Ideally, you'll store your ssh keys on an encrypted volume and protect your private key with a password when prompted for one by `ssh-keygen`. See also "Why do passphrases matter" at https://help.github.com/articles/generating-ssh-keys
- public key: `~/.ssh/id_rsa.pub` - After you've created your ssh keys, add the public key to your GitHub account.

Git on Mac

On a Mac, you won't have git installed unless you have "Command Line Developer Tools" installed but running `git clone` for the first time will prompt you to install them.

Automation of Custom Build Number on Webpage

You can create symbolic links from `.git/hooks/post-checkout` and `.git/hooks/post-commit` to `scripts/installer/custom-build-number-hook` to let Git automatically update `src/main/java/BuildNumber.properties` for you. This will result in showing branch name and commit id in your test deployment webpages on the bottom right corner next to the version.

When you prefer manual updates, there is another script, see above: *Make a Small Change to the Code*.

An alternative to that is using *MicroProfile Config* and set the option `dataverse.build` via a system property, environment variable (`DATAVERSE_BUILD`) or one of the other config sources.

You could even override the version itself with the option `dataverse.version` in the same way, which is usually picked up from a build time source.

See also discussion of version numbers in *Run a Build to Create the War File*.

8.4.10 Sample Data

You may want to populate your **non-production** Dataverse installations with sample data.

https://github.com/IQSS/dataverse-sample-data includes several common data types, data subsetted from production datasets in dataverse.harvard.edu, datasets with file hierarchy, and more.

8.4.11 Switching from Glassfish to Payara

If you already have a working dev environment with Glassfish and want to switch to Payara, you must do the following:

- Copy the "domain1" directory from Glassfish to Payara.

8.4.12 UI Pages Development

While most of the information in this guide focuses on service and backing beans ("the back end") development in Java, working on JSF/Primefaces xhtml pages presents its own unique challenges.

Avoiding Inefficiencies in JSF Render Logic

It is important to keep in mind that the expressions in JSF `rendered=` attributes may be evaluated **multiple** times. So it is crucial not to use any expressions that require database lookups, or otherwise take any appreciable amount of time and resources. Render attributes should exclusively contain calls to methods in backing beans or caching service wrappers that perform any real work on the first call only, then keep returning the cached result on all the consecutive calls. This way it is irrelevant how many times PrimeFaces may need to call the method as any effect on the performance will be negligible.

If you are ever in doubt as to how many times the method in your render logic expression is called, you can simply add a logging statement to the method in question. Or you can simply err on the side of assuming that it's going to be called a lot, and ensure that any repeated calls are not expensive to process.

A simplest, trivial example would be a direct call to a method in SystemConfig service bean. For example,

`<h:outputText rendered="#{systemConfig.advancedModeEnabled}" ...`

If this method (`public boolean isAdvancedModeEnabled()` in `SystemConfig.java`) consults a database setting every time it is called, this database query will be repeated every time JSF reevaluates the expression above. A lookup of a single database setting is not very expensive of course, but repeated enough times unnecessary queries do add up, especially on a busy server. So instead of SystemConfig, SettingsWrapper (a ViewScope bean) should be used to cache the result on the first call:

`<h:outputText rendered="#{settingsWrapper.advancedModeEnabled}" ...`

with the following code in `SettingsWrapper.java`:

```
private Boolean  advancedModeEnabled = null;

public boolean isAdvancedModeEnabled() {
   if (advancedModeEnabled == null) {
      advancedModeEnabled = systemConfig.isAdvancedModeEnabled();
   }
   return advancedModeEnabled;
}
```

A more serious example would be direct calls to PermissionServiceBean methods used in render logic expressions. This is something that has happened and caused some problems in real life. A simple permission service lookup (for example, whether a user is authorized to create a dataset in the current dataverse) can easily take 15 database queries. Repeated multiple times, this can quickly become a measurable delay in rendering the page. PermissionsWrapper must be used exclusively for any such lookups from JSF pages.

See also *Performance*.

JSF1103 Errors

Errors of the form `JSF1103: The metadata facet must be a direct child of the view in viewId / dataverse.xhtml` come from use of the f:metadata tag at the wrong depth in the .xhtml.

Most/all known instances of the problem were corrected in https://github.com/IQSS/dataverse/pull/11128.

Any page that used <ui:composition template="/dataverse_template.xhtml"> was including the f:metadata farther down in the tree rather than as a direct child of the view. As of Payara 6.2025.2, it is not clear that this error was resulting in changes to UI behavior, but the error messages were in the log. If you see these errors, this note and the examples in the PR will hopefully provide some insight as to how to fix them.

8.5 Troubleshooting

Over in the *Classic Dev Environment* section we described the "happy path" of when everything goes right as you set up your Dataverse Software development environment. Here are some common problems and solutions for when things go wrong.

Contents:

- *context-root in payara-web.xml Munged by Netbeans*
- *Configuring / Troubleshooting Mail Host*
- *Rebuilding Your Dev Environment*
- *DataCite*

8.5.1 context-root in payara-web.xml Munged by Netbeans

For unknown reasons, Netbeans will sometimes change the following line under `src/main/webapp/WEB-INF/payara-web.xml`:

```
<context-root>/</context-root>
```

Sometimes Netbeans will change `/` to `/dataverse`. Sometimes it will delete the line entirely. Either way, you will see very strange behavior when attempting to click around the Dataverse installation in a browser. The homepage will load but icons will be missing. Any other page will fail to load entirely and you'll see an app server error.

The solution is to put the file back to how it was before Netbeans touched it. If anyone knows of an open Netbeans bug about this, please let us know.

8.5.2 Configuring / Troubleshooting Mail Host

If you have trouble with the SMTP server, consider editing the `install` script to disable the SMTP check.

Out of the box, no emails will be sent from your development environment. This is because you have to set the `:SystemEmail` setting and make sure you've configured your SMTP server correctly.

You can configure `:SystemEmail` like this:

```
curl -X PUT -d 'Davisverse SWAT Team <davisthedog@harvard.edu>' http://localhost:8080/
api/admin/settings/:SystemEmail
```

Unfortunately for developers not at Harvard, the installer script gives you by default an SMTP server of `mail.hmdc.harvard.edu` but you can specify an alternative SMTP server when you run the installer.

You can check the current SMTP server with the `asadmin` command:

```
./asadmin get server.resources.mail-resource.mail/notifyMailSession.host
```

This command helps verify what host your domain is using to send mail. Even if it's the correct hostname, you may still need to adjust settings. If all else fails, there are some free SMTP service options available such as Gmail and MailGun. This can be configured from the Payara console or the command line.

1. First, navigate to your Payara admin console: http://localhost:4848
2. From the left-side panel, select **JavaMail Sessions**
3. You should see one session named **mail/notifyMailSession** – click on that.

From this window you can modify certain fields of your Dataverse installation's notifyMailSession, which is the JavaMail session for outgoing system email (such as on user sign up or data publication). Two of the most important fields we need are:

- **Mail Host:** The DNS name of the default mail server (e.g. smtp.gmail.com)
- **Default User:** The username provided to your Mail Host when you connect to it (e.g. johndoe@gmail.com)

Most of the other defaults can safely be left as is. **Default Sender Address** indicates the address that your installation's emails are sent from.

If your user credentials for the SMTP server require a password, you'll need to configure some **Additional Properties** at the bottom.

IMPORTANT: Before continuing, it's highly recommended that your Default User account does NOT use a password you share with other accounts, as one of the additional properties includes entering the Default User's password (without concealing it on screen). For smtp.gmail.com you can safely use an app password or create an extra Gmail account for use with your Dataverse Software development environment.

Authenticating yourself to a Mail Host can be tricky. As an example, we'll walk through setting up our JavaMail Session to use smtp.gmail.com as a host by way of SSL on port 465. Use the Add Property button to generate a blank property for each name/value pair.

Name	Value
mail.smtp.auth	true
mail.smtp.password	[user's (*app*) password*]
mail.smtp.port	465
mail.smtp.socketFactory.port	465
mail.smtp.socketFactory.fallback	false
mail.smtp.socketFactory.class	javax.net.ssl.SSLSocketFactory

***WARNING**: Entering a password here will *not* conceal it on-screen. It's recommended to use an *app password* (for smtp.gmail.com users) or utilize a dedicated/non-personal user account with SMTP server auths so that you do not risk compromising your password.

Save these changes at the top of the page and restart your app server to try it out.

The mail session can also be set from command line. To use this method, you will need to delete your notifyMailSession and create a new one. See the below example:

- Delete: `./asadmin delete-javamail-resource mail/MyMailSession`

- Create (remove brackets and replace the variables inside): `./asadmin create-javamail-resource --mailhost [smtp.gmail.com] --mailuser [test\@test\.com] --fromaddress [test\@test\.com] --property mail.smtp.auth=[true]:mail.smtp.password=[password]:mail.smtp.port=[465]:mail.smtp.socketFactory.port=[465]:mail.smtp.socketFactory.fallback=[false]:mail.smtp.socketFactory.class=[javax.net.ssl.SSLSocketFactory] mail/notifyMailSession`

These properties can be tailored to your own preferred mail service, but if all else fails these settings work fine with Dataverse Software development environments for your localhost.

- If you're seeing a "Relay access denied" error in your app server logs when the Dataverse installation attempts to send an email, double check your user/password credentials for the Mail Host you're using.
- If you're seeing a "Connection refused" / similar error upon email sending, try another port.

As another example, here is how to create a Mail Host via command line for Amazon SES:

- Delete: `./asadmin delete-javamail-resource mail/MyMailSession`
- Create (remove brackets and replace the variables inside): `./asadmin create-javamail-resource --mailhost email-smtp.us-east-1.amazonaws.com --mailuser [test\@test\.com] --fromaddress [test\@test\.com] --transprotocol aws --transprotocolclass com.amazonaws.services.simpleemail.AWSJavaMailTransport --property mail.smtp.auth=true:mail.smtp.user=[aws_access_key]:mail.smtp.password=[aws_secret_key]:mail.transport.protocol=smtp:mail.smtp.port=587:mail.smtp.starttls.enable=true mail/notifyMailSession`

8.5.3 Rebuilding Your Dev Environment

A script called `dev-rebuild.sh` is available that does the following:

- Drops the database.
- Clears our Solr.
- Deletes all data files uploaded by users (assuming you are using the default directory).
- Deploys the war file located in the `target` directory.
- Runs `setup-all.sh` in insecure mode so tests will pass.
- Runs post-install SQL statements.
- Publishes the root Dataverse collection.
- Adjusts permissions on on the root Dataverse collection so tests will pass.

To execute the script, make sure you have built a war file already and then `cd` to the root of the source tree and run `scripts/dev/dev-rebuild.sh`. Feedback on this script is welcome!

8.5.4 DataCite

If you are seeing `Response code: 400, [url] domain of URL is not allowed` it's probably because your `dataverse.siteUrl` JVM option is unset or set to localhost (`-Ddataverse.siteUrl=http://localhost:8080`). You can try something like this:

```
./asadmin delete-jvm-options '-Ddataverse.siteUrl=http\://localhost\:8080'
```

```
./asadmin create-jvm-options '-Ddataverse.siteUrl=http\://demo.dataverse.org'
```

8.6 Version Control

The Dataverse Project uses git for version control and GitHub for hosting. On this page we'll explain where to find the code, our branching strategy, advice on how to make a pull request, and other git tips.

Contents:

- *Where to Find the Dataverse Software Code*
- *Branching Strategy*
  - *Goals*
  - *Branches*
    - *The "master" Branch*
    - *The "develop" Branch*
    - *Feature Branches*
    - *Hotfix Branches*
- *How to Make a Pull Request*
  - *Find or Create a GitHub Issue*
    - *Finding GitHub Issues to Work On*
    - *Creating GitHub Issues to Work On*
  - *Communicate Which Issue You Are Working On*
  - *Create a New Branch Off the develop Branch*
  - *Commit Your Change to Your New Branch*
  - *Writing a Release Note Snippet*
  - *Push Your Branch to GitHub*
  - *Make a Pull Request*
  - *Replace Issue with Pull Request*
  - *Make Sure Your Pull Request Has Been Advanced to Code Review*
  - *Summary of Git commands*
- *How to Resolve Conflicts in Your Pull Request*
- *Adding Commits to a Pull Request from a Fork*

- *Handing a Pull Request from a "develop" Branch*

8.6.1 Where to Find the Dataverse Software Code

The main Dataverse Software code is available at https://github.com/IQSS/dataverse but as explained in the *Introduction* section under "Related Projects", there are many other code bases you can hack on if you wish!

8.6.2 Branching Strategy

Goals

The goals of the Dataverse Software branching strategy are:

- allow for concurrent development
- only ship stable code

We follow a simplified "git flow" model described at https://nvie.com/posts/a-successful-git-branching-model/ involving a "master" branch, a "develop" branch, and feature branches such as "1234-bug-fix".

Branches

The "master" Branch

The "master" branch represents released versions of the Dataverse Software. As mentioned in the *Making Releases* section, at release time we update the master branch to include all the code for that release. Commits are never made directly to master. Rather, master is updated only when we merge code into it from the "develop" branch.

The "develop" Branch

The "develop" branch represents code that was stable enough to merge from a "feature" branch (described below) and that will make it into the next release. Like master, commits are never made to the develop branch. The develop branch is where integration occurs. Your goal is have your code merged into the develop branch after it has been reviewed.

Feature Branches

Feature branches are used for both developing features and fixing bugs. They are named after the GitHub issue they are meant to address, so create a GitHub issue if you need to.

"3728-doc-apipolicy-fix" is an example of a fine name for your feature branch. It tells us that you are addressing https://github.com/IQSS/dataverse/issues/3728 and the "slug" is short, descriptive, and starts with the issue number.

Hotfix Branches

Hotfix branches are described under *Making Releases*.

8.6.3 How to Make a Pull Request

Pull requests take all shapes and sizes, from a one-character typo fix to hundreds of files changing at once. Generally speaking, smaller pull requests are better so that they are easier to code review. That said, don't hold back on writing enough code or documentation to address the issue to the best of your ability.

If you are writing code (rather than documentation), please see *Testing* for guidance on writing tests.

The example of creating a pull request below has to do with fixing an important issue with the documentation but applies to fixing code as well.

Find or Create a GitHub Issue

An issue represents a bug (unexpected behavior) or a new feature in Dataverse. We'll use the issue number in the branch we create for our pull request.

Finding GitHub Issues to Work On

Assuming this is your first contribution to Dataverse, you should start with something small. The following issue labels might be helpful in your search:

- good first issue (these appear at https://github.com/IQSS/dataverse/contribute)
- hacktoberfest
- Help Wanted: Code
- Help Wanted: Documentation

For guidance on which issue to work on, please ask! *Getting Help* explains how to get in touch.

Creating GitHub Issues to Work On

You are very welcome to create a GitHub issue to work on. However, for significant changes, please reach out (see *Getting Help*) to make sure the team and community agree with the proposed change.

For small changes and especially typo fixes, please don't worry about reaching out first.

Communicate Which Issue You Are Working On

In the issue you can simply leave a comment to say you're working on it.

If you tell us your GitHub username we are happy to add you to the "read only" team at https://github.com/orgs/IQSS/teams/dataverse-readonly/members so that we can assign the issue to you while you're working on it. You can also tell us if you'd like to be added to the Dataverse Community Contributors spreadsheet.

Create a New Branch Off the develop Branch

Always create your feature branch from the latest code in develop, pulling the latest code if necessary. As mentioned above, your branch should have a name like "3728-doc-apipolicy-fix" that starts with the issue number you are addressing (e.g. #3728) and ends with a short, descriptive name. Dashes ("-") and underscores ("_") in your branch name are ok, but please try to avoid other special characters such as ampersands ("&") that have special meaning in Unix shells. Please do not call your branch "develop" as it can cause maintainers *trouble*.

Commit Your Change to Your New Branch

For each commit to that branch, try to include the issue number along with a summary in the first line of the commit message, such as `Fixed BlockedApiPolicy #3728`. You are welcome to write longer descriptions in the body as well!

Writing a Release Note Snippet

We highly value your insight as a contributor when in comes to describing your work in our release notes. Not every pull request will be mentioned in release notes but most are.

As described at *Write Release Notes*, at release time we compile together release note "snippets" into the final release notes.

Here's how to add a release note snippet to your pull request:

- Create a Markdown file under `doc/release-notes`. You can reuse the name of your branch and append ".md" to it, e.g. `3728-doc-apipolicy-fix.md`
- Edit the snippet to include anything you think should be mentioned in the release notes. Please include the following if they apply:
  - Descriptions of new features or bug fixed, including a link to the HTML preview of the docs you wrote (e.g. https://dataverse-guide–9939.org.readthedocs.build/en/9939/installation/config.html#smtp-email-configuration) and the phrase "For more information, see #3728" (the issue number). If you know the PR number, you can add that too.
  - New configuration settings
  - Upgrade instructions
  - Etc.

Release note snippets do not need to be long. For a new feature, a single line description might be enough. Please note that your release note will likely be edited (expanded or shortened) when the final release notes are being created.

Push Your Branch to GitHub

Push your feature branch to your fork of the Dataverse Software. Your git command may look something like `git push origin 3728-doc-apipolicy-fix`.

Make a Pull Request

Make a pull request to get approval to merge your changes into the develop branch.

Feedback on the pull request template we use is welcome!

Here's an example of a pull request for issue #9729: https://github.com/IQSS/dataverse/pull/10474

Replace Issue with Pull Request

If the pull request closes an issue that has been prioritized, someone from the core team will do the following:

- Move the open issue to the "Done" column of the project board. We do this to track only one card, the pull request, on the project board. Merging the pull request will close the issue because we use the "closes #1234" keyword .
- Copy all labels from the issue to the pull request with the exception of the "size" label.
- Add a size label to the pull request that reflects the amount of review and QA time needed.
- Move the pull request to the "Ready for Review" column.

Make Sure Your Pull Request Has Been Advanced to Code Review

Now that you've made your pull request, your goal is to make sure it appears in the "Code Review" column on the project board.

Look at *Getting Help* for various ways to reach out to developers who have enough access to the GitHub repo to move your issue and pull request to the "Code Review" column.

Summary of Git commands

This section provides sequences of Git commands for three scenarios:

- preparing the first request, when the IQSS Dataverse Software repository and the forked repository are identical
- creating an additional request after some time, when the IQSS Dataverse Software repository is ahead of the forked repository
- while your pull requests are in review the develop branch has been updated, so you have to keep your code base synchronized with the current state of develop branch

In the examples we use 123-COOL-FEATURE as the name of the feature branch, and https://github.com/YOUR_NAME/dataverse.git as your forked repository's URL. In practice modify both accordingly.

1st scenario: preparing the first pull request

```
# clone Dataverse at Github.com ... then

git clone https://github.com/YOUR_NAME/dataverse.git dataverse_fork
cd dataverse_fork

# create a new branch locally for the pull request
git checkout -b 123-COOL-FEATURE

# working on the branch ... then commit changes
git commit -am "#123 explanation of changes"
```

(continues on next page)

(continued from previous page)

```
# upload the new branch to https://github.com/YOUR_NAME/dataverse
git push -u origin 123-COOL-FEATURE

# ... then create pull request at github.com/YOUR_NAME/dataverse
```

2nd scenario: preparing another pull request some month later

```
# register IQSS Dataverse repo
git remote add upstream https://github.com/IQSS/dataverse.git

git checkout develop

# update local develop branch from https://github.com/IQSS/dataverse
git fetch upstream develop
git rebase upstream/develop

# update remote develop branch at https://github.com/YOUR_NAME/dataverse
git push

# create a new branch locally for the pull request
git checkout -b 123-COOL-FEATURE

# work on the branch and commit changes
git commit -am "#123 explanation of changes"

# upload the new branch to https://github.com/YOUR_NAME/dataverse
git push -u origin 123-COOL-FEATURE

# ... then create pull request at github.com/YOUR_NAME/dataverse
```

3rd scenario: synchronize your branch with develop branch

```
git checkout develop

# update local develop branch from https://github.com/IQSS/dataverse
git fetch upstream develop
git rebase upstream/develop

# update remote develop branch at https://github.com/YOUR_NAME/dataverse
git push

# change to the already existing feature branch
git checkout 123-COOL-FEATURE

# merge changes of develop to the feature branch
git merge develop

# check if there are conflicts, if there are follow the next command, otherwise skip to␣
↪next block
# 1. fix the relevant files (including testing)
# 2. commit changes
```

(continues on next page)

(continued from previous page)

```
git add <fixed files>
git commit

# update remote feature branch at https://github.com/YOUR_NAME/dataverse
git push
```

8.6.4 How to Resolve Conflicts in Your Pull Request

Unfortunately, pull requests can quickly become "stale" and unmergable as other pull requests are merged into the develop branch ahead of you. This is completely normal, and often occurs because other developers made their pull requests before you did.

The Dataverse Project team may ping you to ask you to merge the latest from the develop branch into your branch and resolve merge conflicts. If this sounds daunting, please just say so and we will assist you.

If you'd like to resolve the merge conflicts yourself, here are some steps to do so that make use of GitHub Desktop and Netbeans.

In GitHub Desktop:

1. Sync from develop.
2. Open the specific branch that's having the merge conflict.
3. Click "Update from develop".

In Netbeans:

4. Click Window -> Favorites and open your local Dataverse Software project folder in the Favorites panel.
5. In this file browser, you can follow the red cylinder icon to find files with merge conflicts.
6. Double click the red merge conflicted file.
7. Right click on the red tab for that file and select Git -> Resolve Conflicts.
8. Resolve on right or left (if you select "both" you can do finer edits after).
9. Save all changes

In GitHub Desktop:

10. Commit the merge (append issue number to end, e.g. #3728) and leave note about what was resolved.

In GitHub Issues:

11. Leave a comment for the Dataverse Project team that you have resolved the merge conflicts.

8.6.5 Adding Commits to a Pull Request from a Fork

By default, when a pull request is made from a fork, "Allow edits from maintainers" is checked as explained at https://help.github.com/articles/allowing-changes-to-a-pull-request-branch-created-from-a-fork/

This is a nice feature of GitHub because it means that the core dev team for the Dataverse Project can make small (or even large) changes to a pull request from a contributor to help the pull request along on its way to QA and being merged.

GitHub documents how to make changes to a fork at https://help.github.com/articles/committing-changes-to-a-pull-request-branch-created-from-a-fork/ but as of this writing the steps involve making a new clone of the repo. This works but you might find it more convenient to add a "remote" to your existing clone.

The example below uses the fork at https://github.com/uncch-rdmc/dataverse and the branch `4709-postgresql_96` but the technique can be applied to any fork and branch:

```
git remote add uncch-rdmc git@github.com:uncch-rdmc/dataverse.git
git fetch uncch-rdmc
git checkout 4709-postgresql_96
vim path/to/file.txt
git commit
git push uncch-rdmc 4709-postgresql_96
```

8.6.6 Handing a Pull Request from a "develop" Branch

Note: this is something only maintainers of Dataverse need to worry about, typically.

From time to time a pull request comes in from a fork of Dataverse that uses "develop" as the branch behind the PR. (We've started asking contributors not to do this. See *Create a New Branch Off the develop Branch*.) This is problematic because the "develop" branch is the main integration branch for the project. (See *The "develop" Branch*.)

If the PR is perfect and can be merged as-is, no problem. Just merge it. However, if you would like to push commits to the PR, you are likely to run into trouble with multiple "develop" branches locally.

The following is a list of commands oriented toward the simple task of merging the latest from the "develop" branch into the PR but the same technique can be used to push other commits to the PR as well. In this example the PR is coming from username "coder123" on GitHub. At a high level, what we're doing is working in a safe place (/tmp) away from our normal copy of the repo. We clone the main repo from IQSS, check out coder123's version of "develop" (called "dev2" or "false develop"), merge the real "develop" into it, and push to the PR.

If there's a better way to do this, please get in touch!

```
# do all this in /tmp away from your normal code
cd /tmp
git clone git@github.com:IQSS/dataverse.git
cd dataverse
git remote add coder123 git@github.com:coder123/dataverse.git
git fetch coder123
# check out coder123's "develop" to a branch with a different name ("dev2")
git checkout coder123/develop -b dev2
# merge IQSS "develop" into coder123's "develop" ("dev2")
git merge origin/develop
# delete the IQSS "develop" branch locally (!)
git branch -d develop
# checkout "dev2" (false "develop") as "develop" for now
git checkout -b develop
# push the false "develop" to coder123's fork (to the PR)
git push coder123 develop
cd ..
# delete the tmp space (done! \o/)
rm -rf /tmp/dataverse
```

8.7 SQL Upgrade Scripts

The database schema for the Dataverse Software is constantly evolving and we have adopted a tool called Flyway to help keep your development environment up to date and in working order. As you make changes to the database schema (changes to `@Entity` classes), you must write SQL upgrade scripts when needed and follow Flyway file naming conventions.

Contents:

- *Location of SQL Upgrade Scripts*
- *How to Determine if You Need to Create a SQL Upgrade Script*
- *How to Create a SQL Upgrade Script*
- *Troubleshooting*
  - *Renaming SQL Upgrade Scripts*

8.7.1 Location of SQL Upgrade Scripts

`src/main/resources/db/migration` is the directory where we keep SQL upgrade scripts for Flyway to find.

In the past (before adopting Flyway) we used to keep SQL upgrade scripts in `scripts/database/upgrades`. These scripts can still be used as reference but no new scripts should be added there.

8.7.2 How to Determine if You Need to Create a SQL Upgrade Script

If you are creating a new database table (which maps to an `@Entity` in JPA), you do not need to create or update a SQL upgrade script. The reason for this is that we use `create-tables` in `src/main/resources/META-INF/persistence.xml` so that new tables are automatically created by the app server when you deploy your war file.

If you are doing anything other than creating a new database table such as adding a column to an existing table, you must create or update a SQL upgrade script.

8.7.3 How to Create a SQL Upgrade Script

We assume you have already read the *Version Control* section and have been keeping your feature branch up to date with the "develop" branch.

Create a new SQL file in the `src/main/resources/db/migration` directory and put a short, meaningful comment at the top. Make the filename something like `V6.1.0.1.sql`. In this example `6.1` represents the current version of Dataverse, with the last digit representing number of the script for that version. (The zero in this example is a placeholder in case the current version has a third number like `6.1.1`.) Should a newer version be merged while you work on your pull request (PR), you must update your script to the next available number such as `V6.1.0.2.sql`.

Previously, we used longer, more descriptive file naming conventions supported by Flyway. However, this approach occasionally led to inadvertent merging of multiple scripts with the same version, such as `V6.0.0.1__0000-wonderful-pr.sql` and `V6.0.0.1__0001-lovely-pr.sql` where `V6.0.0.1` must be unique. After careful consideration, we agreed to adopt the convention mentioned above for naming files. This helps us detect conflicts before merging a PR, preventing the develop branch from being undeployable due to a Flyway conflict. For more information on Flyway file naming conventions, see https://documentation.red-gate.com/fd/migrations-184127470.html

The SQL migration script you wrote will be part of the war file and executed when the war file is deployed. To see a history of Flyway database migrations that have been applied, look at the `flyway_schema_history` table.

As with any task related to the development of the Dataverse Software, if you need any help writing SQL upgrade scripts, please reach out using any of the channels mentioned under "Getting Help" in the *Introduction* section.

8.7.4 Troubleshooting

Renaming SQL Upgrade Scripts

Please note that if you need to rename your script (because a new version of the Dataverse Software was released, for example), you will see the error "FlywayException: Validate failed: Detected applied migration not resolved locally" when you attempt to deploy and deployment will fail.

To resolve this problem, delete the old migration from the `flyway_schema_history` table and attempt to redeploy.

8.8 Testing

In order to keep our codebase healthy, the Dataverse Project encourages developers to write automated tests in the form of unit tests and integration tests. We also welcome ideas for how to improve our automated testing.

Contents:

- *The Health of a Codebase*
- *Testing in Depth*
- *Unit Tests*
  - *Unit Test Automation Overview*
  - *Writing Unit Tests with JUnit*
    - *Refactoring Code to Make It Unit-Testable*
    - *Parameterized Tests*
    - *JUnit 5 Test Helper Extensions*
    - *Observing Changes to Code Coverage*
    - *Testing Commands*
    - *Running Non-Essential (Excluded) Unit Tests*
- *Integration Tests*
  - *Running the Full API Test Suite Using EC2*
  - *Running the Full API Test Suite Using Docker*
  - *Running the APIs Without Docker (Classic Dev Env)*
    - *The Burrito Key*
    - *Root Collection Permissions*
    - *Publish Root Collection*

        - *dataverse.siteUrl*
        - *dataverse.oai.server.maxidentifiers*
        - *dataverse.oai.server.maxrecords*
        - *Identifier Generation*
    - *Writing API Tests with REST Assured*
    - *Writing and Using a Testcontainers Test*
- *Measuring Coverage of API Tests*
    - *Add jacocoagent.jar to Payara*
    - *Add jacococli.jar to the WAR File*
    - *Deploy the Instrumented WAR File*
    - *Run API Tests to Determine Code Coverage*
    - *Create Code Coverage Report*
    - *Read Code Coverage Report*
- *Load/Performance Testing*
    - *Locust*
    - *download-files.sh script*
- *Continuous Integration*
    - *Enhance build time by caching dependencies*
- *Accessibility Testing*
    - *Accessibility Policy*
    - *Accessibility Tools*
- *Browser-Based Testing*
    - *Browser-Based Testing of JSF*
    - *Browser-Based Testing of the SPA*
- *Future Work*
    - *Future Work on Unit Tests*
    - *Future Work on Integration Tests*
    - *Installation Testing*
    - *Future Work on Load/Performance Testing*
    - *Future Work on Accessibility Testing*

8.8.1 The Health of a Codebase

Before we dive into the nut and bolts of testing, let's back up for a moment and think about why we write automated tests in the first place. Writing automated tests is an investment and leads to better quality software. Counterintuitively, writing tests and executing them regularly allows a project to move faster. Martin Fowler explains this well while talking about the health of a codebase:

> "This is an economic judgment. Several times, many times, I run into teams that say something like, 'Oh well. Management isn't allowing us to do a quality job here because it will slow us down. And we've appealed to management and said we need to put more quality in the code, but they've said no, we need to go faster instead.' And my comment to that is well, as soon as you're framing it in terms of code quality versus speed, you've lost. Because the whole point of refactoring is to go faster.
>
> "And this is why I quite like playing a bit more with the metaphor as the health of a codebase. If you keep yourself healthy then you'll be able to run faster. But if you just say, 'Well, I want to run a lot so I'm therefore going to run a whole load all the time and not eat properly and not pay attention about this shooting pain going up my leg,' then you're not going to be able to run quickly very long. **You have to pay attention to your health. And same with the codebase. You have to continuously say, 'How do we keep it in a healthy state? Then we can go fast,' because we're running marathons here with codebases. And if we neglect that internal quality of the codebase, it hits you surprisingly fast.**"
>
> —Martin Fowler at
> https://devchat.tv/ruby-rogues/178-rr-book-club-refactoring-ruby-with-martin-fowler

8.8.2 Testing in Depth

Security in depth might mean that your castle has a moat as well as high walls. Likewise, when testing, you should consider testing a various layers of the stack using both unit tests and integration tests.

When writing tests, you may find it helpful to first map out which functions of your code you want to test, and then write a functional unit test for each which can later comprise a larger integration test.

8.8.3 Unit Tests

Creating unit tests for your code is a helpful way to test what you've built piece by piece.

Unit tests can be executed without runtime dependencies on PostgreSQL, Solr, or any other external system. They are the lowest level of testing and are executed constantly on developers' laptops as part of the build process and via continuous integration services in the cloud.

A unit test should execute an operation of your code in a controlled fashion. You must make an assertion of what the expected response gives back. It's important to test optimistic output and assertions (the "happy path"), as well as unexpected input that leads to failure conditions. Know how your program should handle anticipated errors/exceptions and confirm with your test(s) that it does so properly.

Unit Test Automation Overview

We use a variety of tools to write, execute, and measure the code coverage of unit tests, including Maven, JUnit, Jacoco, GitHub, and Coveralls. We'll explain the role of each tool below, but here's an overview of what you can expect from the automation we've set up.

As you prepare to make a pull request, as described in the *Version Control* section, you will be working on a new branch you create from the "develop" branch. Let's say your branch is called `1012-private-url`. As you work, you are constantly invoking Maven to build the war file. When you do a "clean and build" in Netbeans, Maven runs all the unit tests (anything ending with `Test.java`) and then runs the results through a tool called Jacoco that calculates code coverage. When you push your branch to GitHub and make a pull request, GitHub Actions runs Maven and Jacoco on your branch and pushes the results to Coveralls, which is a web service that tracks changes to code coverage over time. Note that we have configured Coveralls to not mark small decreases in code coverage as a failure. You can find the Coveralls reports at https://coveralls.io/github/IQSS/dataverse

The main takeaway should be that we care about unit testing enough to measure the changes to code coverage over time using automation. Now let's talk about how you can help keep our code coverage up by writing unit tests with JUnit.

Writing Unit Tests with JUnit

We are aware that there are newer testing tools such as TestNG, but we use JUnit because it's tried and true. We support JUnit 5 based testing and require new tests written with it. (Since Dataverse 6.0, we migrated all of our tests formerly based on JUnit 4.)

If writing tests is new to you, poke around existing unit tests which all end in `Test.java` and live under `src/test`. Each test is annotated with `@Test` and should have at least one assertion which specifies the expected result. In Netbeans, you can run all the tests in it by clicking "Run" -> "Test File". From the test file, you should be able to navigate to the code that's being tested by right-clicking on the file and clicking "Navigate" -> "Go to Test/Tested class". Likewise, from the code, you should be able to use the same "Navigate" menu to go to the tests.

NOTE: Please remember when writing tests checking possibly localized outputs to check against `en_US.UTF-8` and `UTC` l10n strings!

Refactoring Code to Make It Unit-Testable

Existing code is not necessarily written in a way that lends itself to easy testing. Generally speaking, it is difficult to write unit tests for both JSF "backing" beans (which end in `Page.java`) and "service" beans (which end in `Service.java`) because they require the database to be running in order to test them. If service beans can be exercised via API they can be tested with integration tests (described below) but a good technique for making the logic testable it to move code to "util beans" (which end in `Util.java`) that operate on Plain Old Java Objects (POJOs). `PrivateUrlUtil.java` is a good example of moving logic from `PrivateUrlServiceBean.java` to a "util" bean to make the code testable.

Parameterized Tests

Often times you will want to test a method multiple times with similar values. In order to avoid test bloat (writing a test for every data combination), JUnit offers Data-driven unit tests. This allows a test to be run for each set of defined data values.

JUnit 5 offers great parameterized testing. Some guidance how to write those:

- https://junit.org/junit5/docs/current/user-guide/#writing-tests-parameterized-tests
- https://www.baeldung.com/parameterized-tests-junit-5

- https://blog.codefx.org/libraries/junit-5-parameterized-tests/
- See also many examples in our codebase.

Note that JUnit 5 also offers support for custom test parameter resolvers. This enables keeping tests cleaner, as preparation might happen within some extension and the test code is more focused on the actual testing. See https://junit.org/junit5/docs/current/user-guide/#extensions-parameter-resolution for more information.

JUnit 5 Test Helper Extensions

Our codebase provides little helpers to ease dealing with state during tests. Some tests might need to change something which should be restored after the test ran.

For unit tests, the most interesting part is to set a JVM setting just for the current test or a whole test class. (Which might be an inner class, too!). Please make use of the `@JvmSetting(key = JvmSettings.XXX, value = "")` annotation and also make sure to annotate the test class with `@LocalJvmSettings`.

Inspired by JUnit's `@MethodSource` annotation, you may use `@JvmSetting(key = JvmSettings.XXX, method = "zzz")` to reference a static method located in the same test class by name (i. e. `private static String zzz() {}`) to allow retrieving dynamic data instead of String constants only. (Note the requirement for a *static* method!)

If you want to delete a setting, simply provide a `null` value. This can be used to override a class-wide setting or some other default that is present for some reason.

To set arbitrary system properties for the current test, a similar extension `@SystemProperty(key = "", value = "")` has been added. (Note: it does not support method references.)

Both extensions will ensure the global state of system properties is non-interfering for test executions. Tests using these extensions will be executed in serial.

This settings helper may be extended at a later time to manipulate settings in a remote instance during integration or end-to-end testing. Stay tuned!

Observing Changes to Code Coverage

Once you've written some tests, you're probably wondering how much you've helped to increase the code coverage. In Netbeans, do a "clean and build." Then, under the "Projects" tab, right-click "dataverse" and click "Code Coverage" -> "Show Report". For each Java file you have open, you should be able to see the percentage of code that is covered by tests and every line in the file should be either green or red. Green indicates that the line is being exercised by a unit test and red indicates that it is not.

In addition to seeing code coverage in Netbeans, you can also see code coverage reports by opening `target/site/jacoco-X-test-coverage-report/index.html` in your browser. Depending on the report type you want to look at, let `X` be one of `unit`, `integration` or `merged`. "Merged" will display combined coverage of both unit and integration test, but does currently not cover API tests.

Testing Commands

You might find studying the following test classes helpful in writing tests for commands:

- CreatePrivateUrlCommandTest.java
- DeletePrivateUrlCommandTest.java
- GetPrivateUrlCommandTest.java

In addition, there is a writeup on "The Testable Command" at https://github.com/IQSS/dataverse/blob/master/doc/theTestableCommand/TheTestableCommand.md .

Running Non-Essential (Excluded) Unit Tests

You should be aware that some unit tests have been deemed "non-essential" and have been annotated with `@Tag(Tags.NOT_ESSENTIAL_UNITTESTS)` and are excluded from the "dev" Maven profile, which is the default profile. All unit tests (that have not been annotated with `@Disable`), including these non-essential tests, are run from continuous integration systems such as Jenkins and GitHub Actions with the following `mvn` command that invokes a non-default profile:

```
mvn test -P all-unit-tests
```

Generally speaking, unit tests have been flagged as non-essential because they are slow or because they require an Internet connection. You should not feel obligated to run these tests continuously but you can use the `mvn` command above to run them. To iterate on the unit test in Netbeans and execute it with "Run -> Test File", you must temporarily comment out the annotation flagging the test as non-essential.

8.8.4 Integration Tests

Unit tests are fantastic for low level testing of logic but aren't especially real-world-applicable because they do not exercise the Dataverse Software as it runs in production with a database and other runtime dependencies. We test in-depth by also writing integration tests to exercise a running system.

Unfortunately, the term "integration tests" can mean different things to different people. For our purposes, an integration test can have two flavors:

1. Be an API Test:
    - Exercise the Dataverse Software APIs.
    - Running not automatically on developers' laptops.
    - Operate on a Dataverse installation that is running and able to talk to both PostgreSQL and Solr.
    - Written using REST Assured.
2. Be a Testcontainers Test:
    - Operates any dependencies via the Testcontainers API, using containers.
    - Written as a JUnit test, using all things necessary to test.
    - Makes use of the Testcontainers framework.
    - Able to run anywhere having Docker around (podman support under construction).

Running the Full API Test Suite Using EC2

Prerequisite: To run the API test suite in an EC2 instance you should first follow the steps in the *Deployment* section to get set up with the AWS binary to launch EC2 instances. If you're here because you just want to spin up a branch, you'll still want to follow the AWS deployment setup steps, but may find the ec2-create README.md Quick Start section helpful.

You may always retrieve a current copy of the ec2-create-instance.sh script and accompanying group_var.yml file from the dataverse-ansible repo. Since we want to run the test suite, let's grab the group_vars used by Jenkins:

- ec2-create-instance.sh
- jenkins.yml

Edit `jenkins.yml` to set the desired GitHub repo and branch, and to adjust any other options to meet your needs:

- `dataverse_repo: https://github.com/IQSS/dataverse.git`
- `dataverse_branch: develop`
- `dataverse.api.test_suite: true`
- `dataverse.unittests.enabled: true`
- `dataverse.sampledata.enabled: true`

If you wish, you may pass the script a `-l` flag with a local relative path in which the script will copy various logs at the end of the test suite for your review.

Finally, run the script:

```
$ ./ec2-create-instance.sh -g jenkins.yml -l log_dir
```

Running the Full API Test Suite Using Docker

To run the full suite of integration tests on your laptop, we recommend running Dataverse and its dependencies in Docker, as explained in the *Development Usage* section of the Container Guide. This environment provides additional services (such as S3) that are used in testing.

Running the APIs Without Docker (Classic Dev Env)

While it is possible to run a good number of API tests without using Docker in our *Classic Dev Environment*, we are transitioning toward including additional services (such as S3) in our Dockerized development environment (*Development Usage*), so you will probably find it more convenient to it instead.

Unit tests are run automatically on every build, but dev environments and servers require special setup to run API (REST Assured) tests. In short, the Dataverse software needs to be placed into an insecure mode that allows arbitrary users and datasets to be created and destroyed (this is done automatically in the Dockerized environment, as well as the steps described below). This differs greatly from the out-of-the-box behavior of the Dataverse software, which we strive to keep secure for sysadmins installing the software for their institutions in a production environment.

The Burrito Key

For reasons that have been lost to the mists of time, the Dataverse software really wants you to to have a burrito. Specifically, if you're trying to run REST Assured tests and see the error "Dataverse config issue: No API key defined for built in user management", you must run the following curl command (or make an equivalent change to your database):

```
curl -X PUT -d 'burrito' http://localhost:8080/api/admin/settings/:BuiltinUsersKey
```

Without this "burrito" key in place, REST Assured will not be able to create users. We create users to create objects we want to test, such as collections, datasets, and files.

Root Collection Permissions

In your browser, log in as dataverseAdmin (password: admin) and click the "Edit" button for your root collection. Navigate to Permissions, then the Edit Access button. Under "Who can add to this collection?" choose "Anyone with a Dataverse installation account can add sub collections and datasets" if it isn't set to this already.

Alternatively, this same step can be done with this script: `scripts/search/tests/grant-authusers-add-on-root`

Publish Root Collection

The root collection must be published for some of the REST Assured tests to run.

dataverse.siteUrl

When run locally (as opposed to a remote server), some of the REST Assured tests require the `dataverse.siteUrl` JVM option to be set to `http://localhost:8080`. See *JVM Options* section in the Installation Guide for advice changing JVM options. First you should check to check your JVM options with:

```
./asadmin list-jvm-options | egrep 'dataverse|doi'
```

If `dataverse.siteUrl` is absent, you can add it with:

```
./asadmin create-jvm-options "-Ddataverse.siteUrl=http\://localhost\:8080"
```

dataverse.oai.server.maxidentifiers

The OAI Harvesting tests require that the paging limit for ListIdentifiers must be set to 2, in order to be able to trigger this paging behavior without having to create and export too many datasets:

```
./asadmin create-jvm-options "-Ddataverse.oai.server.maxidentifiers=2"
```

dataverse.oai.server.maxrecords

The OAI Harvesting tests require that the paging limit for ListRecords must be set to 2, in order to be able to trigger this paging behavior without having to create and export too many datasets:

```
./asadmin create-jvm-options "-Ddataverse.oai.server.maxrecords=2"
```

Identifier Generation

`DatasetsIT.java` exercises the feature where the "identifier" of a DOI can be a digit and requires a sequence to be added to your database. See `:IdentifierGenerationStyle` under the *Configuration* section for adding this sequence to your installation of PostgreSQL.

Writing API Tests with REST Assured

Before writing any new REST Assured tests, you should get the tests to pass in an existing REST Assured test file. `BuiltinUsersIT.java` is relatively small and requires less setup than other test files.

You do not have to reinvent the wheel. There are many useful methods you can call in your own tests – especially within UtilIT.java – when you need your test to create and/or interact with generated accounts, files, datasets, etc. Similar methods can subsequently delete them to get them out of your way as desired before the test has concluded.

For example, if you're testing your code's operations with user accounts, the method `UtilIT.createRandomUser();` can generate an account for your test to work with. The same account can then be deleted by your program by calling the `UtilIT.deleteUser();` method on the imaginary friend your test generated.

Remember, it's only a test (and it's not graded)! Some guidelines to bear in mind:

- Map out which logical functions you want to test
- Understand what's being tested and ensure it's repeatable
- Assert the conditions of success / return values for each operation * A useful resource would be HTTP status codes
- Let the code do the labor; automate everything that happens when you run your test file.
- If you need to test an optional service (S3, etc.), add it to our docker compose file. See *Development Usage*.
- Just as with any development, if you're stuck: ask for help!

To execute existing integration tests on your local Dataverse installation from the command line, use Maven. You should have Maven installed as per *Development Environment*, but if not it's easily done via Homebrew: `brew install maven`.

Once installed, you may run commands with `mvn [options] [<goal(s)>] [<phase(s)>]`.

- If you want to run just one particular API test class:

  ```
  mvn test -Dtest=UsersIT
  ```

- If you want to run just one particular API test method,

  ```
  mvn test -Dtest=UsersIT#testMergeAccounts
  ```

- To run more than one test at a time, separate by commas:

  ```
  mvn test -Dtest=FileRecordJobIT,ConfirmEmailIT
  ```

- To run any test(s) on a particular domain, replace localhost:8080 with desired domain name:

  ```
  mvn test -Dtest=FileMetadataIT -Ddataverse.test.baseurl='http://localhost:8080'
  ```

- Solr only makes new documents visible after its next soft commit, so the tests wait `solr.autoSoftCommit.maxTime` (1000 ms by default) after indexing. The Docker dev stack sets it to 100 ms; tell the tests to match it and they run faster:

  ```
  mvn test -Dtest=SearchIT -Ddataverse.test.solr.softcommit.millis=100
  ```

If you are adding a new test class, be sure to add it to `tests/integration-tests.txt` so that our automated testing knows about it.

Writing and Using a Testcontainers Test

Most scenarios of integration testing involve having dependent services running. This is where Testcontainers kicks in by providing a JUnit interface to drive them before and after executing your tests.

Test scenarios are endless. Some examples are migration scripts, persistance, storage adapters etc.

To run a test with Testcontainers, you will need to write a JUnit 5 test. The upstream project provides some documentation about this.

Please make sure to:

1. End your test class with `IT`
2. Annotate the test class with two tags:

```
/** A very minimal example for a Testcontainers integration test class. */
@Testcontainers(disabledWithoutDocker = true)
@Tag(edu.harvard.iq.dataverse.util.testing.Tags.INTEGRATION_TEST)
@Tag(edu.harvard.iq.dataverse.util.testing.Tags.USES_TESTCONTAINERS)
class MyExampleIT { /* ... */ }
```

If using upstream modules, e.g. for PostgreSQL or similar, you will need to add a dependency to `pom.xml` if not present. See the PostgreSQL module example.

To run these tests, simply call out to Maven:

```
mvn verify
```

Notes:

1. Remember to have Docker ready to serve or tests will fail.
2. You can skip running unit tests by adding `-DskipUnitTests`
3. You can choose to ignore test with Testcontainers by adding `-Dit.groups='integration & ! testcontainers'` Learn more about filter expressions in the JUnit 5 guide.

8.8.5 Measuring Coverage of API Tests

Measuring the code coverage of API tests with Jacoco requires several steps. In order to make these steps clear we'll use "/usr/local/payara7" as the Payara directory and "dataverse" as the Payara Unix user.

Please note that this was tested under Glassfish 4 but it is hoped that the same steps will work with Payara.

Add jacococoagent.jar to Payara

In order to get code coverage reports out of Payara we'll be adding jacococoagent.jar to the Payara "lib" directory.

First, we need to download Jacoco. Look in pom.xml to determine which version of Jacoco we are using. As of this writing we are using 0.8.1 so in the example below we download the Jacoco zip from https://github.com/jacoco/jacoco/releases/tag/v0.8.1

Note that we are running the following commands as the user "dataverse". In short, we stop Payara, add the Jacoco jar file, and start up Payara again.

```
su - dataverse
cd /home/dataverse
mkdir -p local/jacoco-0.8.1
cd local/jacoco-0.8.1
wget https://github.com/jacoco/jacoco/releases/download/v0.8.1/jacoco-0.8.1.zip
unzip jacoco-0.8.1.zip
/usr/local/payara7/bin/asadmin stop-domain
cp /home/dataverse/local/jacoco-0.8.1/lib/jacocoagent.jar /usr/local/payara7/glassfish/
↪lib
/usr/local/payara7/bin/asadmin start-domain
```

Add jacococli.jar to the WAR File

As the "dataverse" user download `instrument_war_jacoco.bash` (or skip ahead to the "git clone" step to get the script that way) and give it two arguments:

- path to your pristine WAR file
- path to the new WAR file the script will create with jacococli.jar in it

```
./instrument_war_jacoco.bash dataverse.war dataverse-jacoco.war
```

Deploy the Instrumented WAR File

Please note that you'll want to undeploy the old WAR file first, if necessary.

Run this as the "dataverse" user.

```
/usr/local/payara7/bin/asadmin deploy dataverse-jacoco.war
```

Note that after deployment the file "/usr/local/payara7/glassfish/domains/domain1/config/jacoco.exec" exists and is empty.

Run API Tests to Determine Code Coverage

Note that if you are looking for how to run API tests generally, you should refer to *Integration Tests*.

Note that "/usr/local/payara7/glassfish/domains/domain1/config/jacoco.exec" will become non-empty after you stop and start Payara. You must stop and start Payara before every run of the integration test suite.

```
/usr/local/payara7/bin/asadmin stop-domain
/usr/local/payara7/bin/asadmin start-domain
git clone https://github.com/IQSS/dataverse.git
```

(continues on next page)

(continued from previous page)

```
cd dataverse
TESTS=$(<tests/integration-tests.txt)
mvn test -Dtest=$TESTS
```

(As an aside, you are not limited to API tests for the purposes of learning which code paths are being executed. You could click around the GUI, for example. Jacoco doesn't know or care how you exercise the application.)

Create Code Coverage Report

Run these commands as the "dataverse" user. The `cd dataverse` means that you should change to the directory where you cloned the "dataverse" git repo.

```
cd dataverse
java -jar /home/dataverse/local/jacoco-0.8.1/lib/jacococli.jar report --classfiles␣
↪target/classes --sourcefiles src/main/java --html target/coverage-it/ /usr/local/
↪payara7/glassfish/domains/domain1/config/jacoco.exec
```

Read Code Coverage Report

target/coverage-it/index.html is the place to start reading the code coverage report you just created.

8.8.6 Load/Performance Testing

Locust

Load and performance testing is conducted on an as-needed basis but we're open to automating it. As of this writing Locust (https://locust.io) scripts at https://github.com/IQSS/dataverse-helper-scripts/tree/master/src/stress_tests have been used.

download-files.sh script

One way of generating load is by downloading many files. You can download `download-files.sh`, make it executable (`chmod 755`), and run it with `--help`. You can use `-b` to specify the base URL of the Dataverse installation and `-s` to specify the number of seconds to wait between requests like this:

```
./download-files.sh -b https://dev1.dataverse.org -s 2
```

The script requires a file called `files.txt` to operate and database IDs for the files you want to download should each be on their own line.

8.8.7 Continuous Integration

The Dataverse Project currently makes use of two Continuous Integration platforms, Jenkins and GitHub Actions.

Our Jenkins config is a work in progress and may be viewed at https://github.com/IQSS/dataverse-jenkins/ A corresponding GitHub webhook is required. Build output is viewable at https://jenkins.dataverse.org/

GitHub Actions jobs can be found in `.github/workflows`.

As always, pull requests to improve our continuous integration configurations are welcome.

Enhance build time by caching dependencies

In the future, CI builds in ephemeral build environments and Docker builds can benefit from caching all dependencies and plugins. As the Dataverse Project is a huge project, build times can be enhanced by avoiding re-downloading everything when the Maven POM is unchanged. To seed the cache, use the following Maven goal before using Maven in (optional) offline mode in your scripts:

```
mvn de.qaware.maven:go-offline-maven-plugin:resolve-dependencies``
mvn -o package -DskipTests
```

The example above builds the WAR file without running any tests. For other scenarios: not using offline mode allows Maven to download more dynamic dependencies, which are not easy to track, like Surefire Plugins. Overall downloads will reduced anyway.

You will obviously have to utilize caching functionality of your CI service or do proper Docker layering.

8.8.8 Accessibility Testing

Accessibility Policy

The Dataverse Project aims to improve the user experience for those with disabilities, and are in the process of following the recommendations of the Harvard University Digital Accessibility Policy, which use the Worldwide Web Consortium's Web Content Accessibility Guidelines version 2.1, Level AA Conformance (WCAG 2.1 Level AA) as the standard.

To report an accessibility issue with the Dataverse Software, you can create a new issue in our GitHub repo at: https://github.com/IQSS/dataverse/issues/

Accessibility Tools

Our development process will incorporate automated testing provided by tools like SiteImprove and Accessibility Management Platform (AMP) from Level Access, to run accessibility reports for the application.

Developers who contribute front-end UI code are responsible for understanding the requirements of this standard and the tools and methods for securing conformance with it.

There are browser developer tools such as the Wave toolbar by WebAIM (available for Chrome, Firefox) and the Siteimprove Accessibility Checker (available for Chrome, Firefox) that will generate reports for a single page. It is required that developers utilize these tools to catch any accessibility issues with pages or features that are being added to the application UI.

8.8.9 Browser-Based Testing

Browser-Based Testing of JSF

The JSF frontend is being tested with Playwright via a GitHub Action that runs a test suite from https://github.com/gdcc/dataverse-jsf-tests (see https://github.com/IQSS/dataverse/pull/12526).

Browser-Based Testing of the SPA

https://github.com/IQSS/dataverse-frontend uses Cypress.

8.8.10 Future Work

We'd like to make improvements to our automated testing. See also 'this thread from our mailing list <https://groups.google.com/forum/#!topic/dataverse-community/X8OrRWbPimA>'_ asking for ideas from the community, and discussion at 'this GitHub issue. <https://github.com/IQSS/dataverse/issues/2746>'_

Future Work on Unit Tests

- Review pull requests from @bencomp for ideas for approaches to testing: https://github.com/IQSS/dataverse/pulls?q=is%3Apr+author%3Abencomp
- Come up with a way to test commands: http://irclog.iq.harvard.edu/dataverse/2015-11-04#i_26750
- Test EJBs using Arquillian, embedded app servers, or similar. @bmckinney kicked the tires on Arquillian at https://github.com/bmckinney/bio-dataverse/commit/2f243b1db1ca704a42cd0a5de329083763b7c37a

Future Work on Integration Tests

- Automate testing of dataverse-client-python: https://github.com/IQSS/dataverse-client-python/issues/10
- Work with @leeper on testing the R client: https://github.com/IQSS/dataverse-client-r
- Review and attempt to implement "API Test Checklist" from @kcondon at https://docs.google.com/document/d/199Oq1YwQ4pYCguaeW48bIN28QAitSk63NbPYxJHCCAE/edit?usp=sharing
- Generate code coverage reports for **integration** tests: https://github.com/pkainulainen/maven-examples/issues/3 and https://www.petrikainulainen.net/programming/maven/creating-code-coverage-reports-for-unit-and-integration-tests-with-the-jacoco-maven-plugin/
- Consistent logging of API Tests. Show test name at the beginning and end and status codes returned.
- expected passing and known/expected failing integration tests: https://github.com/IQSS/dataverse/issues/4438

Installation Testing

- Work with @donsizemore to automate testing of https://github.com/gdcc/dataverse-ansible

Future Work on Load/Performance Testing

- Clean up and copy stress tests code, config, and docs into main repo from https://github.com/IQSS/dataverse-helper-scripts/tree/master/src/stress_tests
- Marcel Duran created a command-line wrapper for the WebPagetest API that can be used to test performance in your continuous integration pipeline (TAP, Jenkins, etc.): https://github.com/marcelduran/webpagetest-api/wiki/Test-Specs#jenkins-integration
- Create top-down checklist, building off the "API Test Coverage" spreadsheet at https://github.com/IQSS/dataverse/issues/3358#issuecomment-256400776

Future Work on Accessibility Testing

- Using https://github.com/gdcc/dataverse-ansible and hooks available from accessibility testing tools, automate the running of accessibility tools on PRs so that developers will receive quicker feedback on proposed code changes that reduce the accessibility of the application.

8.9 API Design

API design is a large topic. We expect this page to grow over time.

Contents:

- *OpenAPI*
  - *OpenAPI Annotations*
  - *OpenAPI Quality*
- *Paths*
  - *Exposing Settings*

8.9.1 OpenAPI

As you add API endpoints, please be conscious that we are exposing these endpoints as an OpenAPI document at `/openapi` (e.g. http://localhost:8080/openapi). See *Getting the OpenAPI Document* in the API Guide for the user-facing documentation.

OpenAPI Annotations

To keep our OpenAPI document at a reasonable quality, you must add annotations to API endpoints you are adding and editing. Here's an example:

```
@POST
@AuthRequired
@Path("/")
@Operation(summary = "Creates a license", description = "Creates a license when the␣
↪authenticated user is a superuser and returns the created license location.")
@APIResponse(responseCode = "201", description = "License created with a Location header␣
↪pointing to the new license.")
public Response addLicense(@Context ContainerRequestContext crc,
        @RequestBody(description = "License definition to persist, including name, URI,␣
↪active state, and sort order.",
                content = @Content(mediaType = "application/json",
                        schema = @Schema(type = SchemaType.OBJECT,
                                description = "License definition accepted by the␣
↪license administration API.")))
        License license) {
...
}
```

If you are looking for a reference about the annotations used to generate the OpenAPI document, you can find it in the MicroProfile OpenAPI Specification.

OpenAPI Quality

To use vacuum with our recommended settings to check the quality of our OpenAPI document, see the `README.md` under `scripts/openapi`.

You can prevent additional problems in our OpenAPI document by observing the following practices:

- When creating a method name within an API class, make it unique.
- Check that HTTP GET and DELETE operations do not accept request bodies.
- Various notes under `vacuum-recommended.yaml`.

8.9.2 Paths

A reminder from Wikipedia of what a path is:

```
        userinfo       host      port
        _____ ____________
https://john.doe@www.example.com:123/forum/questions/?tag=networking&order=newest#top
└──   └──────────────────────────┴───────────────┘ └──────────────────────────┘ └
scheme          authority                  path                  query           fragment
```

Exposing Settings

Since Dataverse 4, database settings have been exposed via API at http://localhost:8080/api/admin/settings

(JVM options are probably available via the Payara REST API, but this is out of scope.)

Settings need to be exposed outside to API clients outside of `/api/admin` (which is typically restricted to localhost). Here are some guidelines to follow when exposing settings.

- When you are exposing a database setting as-is:
  - Use `/api/info/settings` as the root path.
  - Append the name of the setting including the colon (e.g. `:DatasetPublishPopupCustomText`)
  - Final path example: `/api/info/settings/:DatasetPublishPopupCustomText`
- If the absence of the database setting is filled in by a default value (e.g. `:ZipDownloadLimit` or `:ApiTermsOfUse`):
  - Use `/api/info` as the root path.
  - Append the setting but remove the colon and downcase the first character (e.g. `zipDownloadLimit`)
  - Final path example: `/api/info/zipDownloadLimit`
- If the database setting you're exposing make more sense outside of `/api/info` because there's more context (e.g. `:CustomDatasetSummaryFields`):
  - Feel free to use a path outside of `/api/info` as the root path.
  - Given additional context, append a shortened name (e.g. `/api/datasets/summaryFieldNames`).
  - Final path example: `/api/datasets/summaryFieldNames`

- If you need to expose a JVM option (MicroProfile setting) such as `dataverse.api.allow-incomplete-metadata`:
  - Use `/api/info` as the root path.
  - Append a meaningful name for the setting (e.g. `incompleteMetadataViaApi`).
  - Final path example: `/api/info/incompleteMetadataViaApi`

8.10 Security

These are security resources for Dataverse developers.

- *User-Facing Security-Related Documentation*
- *Tracking Security Issues*
- *Sending Security Advisories*
- *Giving Credit to Security Researchers*

8.10.1 User-Facing Security-Related Documentation

You should familiarize yourself with the following content we provide for our users to read:

- *Ongoing Security of Your Installation*
- *Security Advisories*
- *Reporting Security Issues*
- SECURITY.md (rendered for the public at https://github.com/IQSS/dataverse/security)

Some information is repeated across these documents and we should keep them in sync.

8.10.2 Tracking Security Issues

We use a private GitHub issue tracker at https://github.com/IQSS/dataverse-security/issues for security issues.

8.10.3 Sending Security Advisories

See our how to send a security advisory doc.

8.10.4 Giving Credit to Security Researchers

We usually give credit to security researchers in release notes. See *Write Release Notes* and *Reporting Security Issues*.

8.11 Performance

Performance is a feature was a mantra when Stack Overflow was being developed. We endeavor to do the same with Dataverse!

In this section we collect ideas and share practices for improving performance.

Contents:

- *Problem Statement*
- *Current Practices*
  - *Cache When You Can*
  - *Use Async*
  - *Use a Queue*
  - *Offload Expensive Operations Outside the App Server*
  - *Drop to Raw SQL as Necessary*
  - *Add Indexes to Database Tables*
  - *Find Bottlenecks with a Profiler*
  - *Warn Developers in Code Comments*
  - *Write Docs for Devs about Perf*
  - *Horizontal Scaling of App Server*
  - *Code Review and QA*
  - *Locust Testing at Release Time*
  - *Issue Tracking and Prioritization*
  - *Document Performance Tools*
  - *Google Analytics*
- *Abandoned Tools and Practices*
  - *New Relic*
- *Areas of Particular Concern*
  - *Command Engine Execution Rate Metering*
  - *Solr*
  - *Datasets with Large Numbers of Files or Versions*
  - *Withstanding Bots*
- *Suggested Practices*

  - *Implement the Frontend Plan for Performance*
  - *Set up Query Counter in Jenkins*
  - *Count Database Queries per API Test*
  - *Teach Developers How to Do Performance Testing Locally*
  - *Automate Performance Testing*
  - *Make Production Data or Equivalent Available to Developers*
  - *Automate Performance Testing with Production Data or Equivalent*
  - *Use Monitoring as Performance Testing*
  - *Learn from Training and Conferences*
  - *Learn from the Community How They Monitor Performance*
  - *Teach the Community to Do Performance Testing*
- *Conclusion*

8.11.1 Problem Statement

Performance has always been important to the Dataverse Project, but results have been uneven. We've seen enough success in the marketplace that performance must be adequate, but internally we sometimes refer to Dataverse as a pig.

8.11.2 Current Practices

We've adopted a number of practices to help us maintain our current level of performance and most should absolutely continue in some form, but challenges mentioned throughout should be addressed to further improve performance.

Cache When You Can

The Metrics API, for example, caches values for 7 days by default. We took a look at JSR 107 (JCache - Java Temporary Caching API) in #2100. We're aware of the benefits of caching.

Use Async

We index datasets (and all objects) asynchronously. That is, we let changes persist in the database and afterward copy the data into Solr.

Use a Queue

We use a JMS queue for when ingesting tabular files. We've talked about adding a queue (even an external queue) for indexing, DOI registration, and other services.

Offload Expensive Operations Outside the App Server

When operations are computationally expensive, we have realized performance gains by offloading them to systems outside of the core code. For example, rather than having files pass through our application server when they are downloaded, we use direct download so that client machines download files directly from S3. (We use the same trick with upload.) When a client downloads multiple files, rather than zipping them within the application server as before, we now have a separate "zipper" process that does this work out of band.

Drop to Raw SQL as Necessary

We aren't shy about writing raw SQL queries when necessary. We've written querycount scripts to help identify problematic queries and mention slow query log at *Monitoring*.

Add Indexes to Database Tables

There was a concerted effort in #1880 to add indexes to a large number of columns, but it's something we're mindful of, generally. Perhaps we could use some better detection of when indexes would be valuable.

Find Bottlenecks with a Profiler

VisualVM is popular and bundled with Netbeans. Many options are available including JProfiler.

Warn Developers in Code Comments

For code that has been optimized for performance, warnings are sometimes inserted in the form of comments for future developers to prevent backsliding.

Write Docs for Devs about Perf

Like this doc. :)

Sometimes perf is written about in other places, such as *Avoiding Inefficiencies in JSF Render Logic*.

Horizontal Scaling of App Server

We've made it possible to run more than one application server, though it requires some special configuration. This way load can be spread out across multiple servers. For details, see *Multiple App Servers* in the Installation Guide.

Code Review and QA

Before code is merged, while it is in review or QA, if a performance problem is detected (usually on an ad hoc basis), the code is returned to the developer for improvement. Developers and reviewers typically do not have many tools at their disposal to test code changes against anything close to production data. QA maintains a machine with a copy of production data but tests against smaller data unless a performance problem is suspected.

A new QA guide is coming in https://github.com/IQSS/dataverse/pull/10103

Locust Testing at Release Time

As one of the final steps in preparing for a release, QA runs performance tests using a tool called Locust as explained the Developer Guide (see *Locust*). The tests are not comprehensive, testing only a handful of pages with anonymous users, but they increase confidence that the upcoming release is not drastically slower than previous releases.

Issue Tracking and Prioritization

Performance issues are tracked in our issue tracker under the Feature: Performance & Stability label (e.g. #7788). That way, we can track performance problems throughout the application. Unfortunately, the pain is often felt by users in production before we realize there is a problem. As needed, performance issues are prioritized to be included in a sprint, to speed up the collection page, for example.

Document Performance Tools

In the *Monitoring* page section of the Admin Guide we describe how to set up Munin for monitoring performance of an operating system. We also explain how to set up Performance Insights to monitor AWS RDS (PostgreSQL as a service, in our case). In the *Tools* section of the Developer Guide, we have documented how to use Eclipse Memory Analyzer Tool (MAT), SonarQube, jmap, and jstat.

Google Analytics

Emails go to a subset of the team monthly with subjects like "Your September Search performance for https://dataverse.harvard.edu" with a link to a report but it's mostly about the number clicks, not how fast the site is. It's unclear if it provides any value with regard to performance.

8.11.3 Abandoned Tools and Practices

New Relic

For many years Harvard Dataverse was hooked up to New Relic, a tool that promises all-in-one observability, according to their website. In practice, we didn't do much with the data.

8.11.4 Areas of Particular Concern

Command Engine Execution Rate Metering

We'd like to rate limit commands (CreateDataset, etc.) so that we can keep them at a reasonable level (#9356). This is similar to how many APIs are rate limited, such as the GitHub API.

Solr

While in the past Solr performance hasn't been much of a concern, in recent years we've noticed performance problems when Harvard Dataverse is under load. Improvements were made in PR #10050, for example.

We are tracking performance problems in #10469.

In a meeting with a Solr expert on 2024-05-10 we were advised to avoid joins as much as possible. (It was acknowledged that many Solr users make use of joins because they have to, like we do, to keep some documents private.) Toward that end we have added two feature flags called *dataverse.feature.avoid-expensive-solr-join* and *dataverse.feature.add-publicobject-solr-field*. It was confirmed experimentally that performing the join on all the public objects (published collections, datasets and files), i.e., the bulk of the content in the search index, was indeed very expensive, especially on a large instance the size of the IQSS prod. archive, especially under indexing load. We confirmed that it was in fact unnecessary and were able to replace it with a boolean field directly in the indexed documents, which is achieved by the two feature flags above. However, as of writing this, this mechanism should still be considered experimental. Another feature flag, *dataverse.feature.reduce-solr-deletes*, avoids deleting Solr documents for files in a dataset prior to sending updates. It also eliminates several causes of orphan permission documents. This is expected to improve indexing performance to some extent and is a step towards avoiding unnecessary updates (i.e. when a doc would not change).

Datasets with Large Numbers of Files or Versions

We'd like to scale Dataverse to better handle large number of files or versions. Progress was made in PR #9883.

Withstanding Bots

Google bot, etc.

8.11.5 Suggested Practices

Many of our current practices should remain in place unaltered. Others could use some refinement. Some new practices should be adopted as well. Here are some suggestions.

Implement the Frontend Plan for Performance

The Dataverse - SPA MVP Definition doc has some ideas around how to achieve good performance for the new front end in the areas of rendering, monitoring,file upload/download, pagination, and caching. We should create as many issues as necessary in the frontend repo and work on them in time. The doc recommends the use of React Profiler and other tools. Not mentioned is https://pagespeed.web.dev but we can investigate it as well. See also #183, a parent issue about performance. In #184 we plan to compare the performance of the old JSF UI vs. the new React UI. Cypress plugins for load testing could be investigated.

Set up Query Counter in Jenkins

See countquery script above. See also https://jenkins.dataverse.org/job/IQSS-dataverse-develop/ws/target/query_count.out

Show the plot over time. Make spikes easily apparent. 320,035 queries as of this writing.

Count Database Queries per API Test

Is it possible? Just a thought.

Teach Developers How to Do Performance Testing Locally

Do developers know how to use a profiler? Should they use JMeter? statsd-jvm-profiler? How do you run our *Locust* tests? Should we continue using that tool? Give developers time and space to try out tools and document any tips along the way. For this stage, small data is fine.

Automate Performance Testing

We are already using two excellent continuous integration (CI) tools, Jenkins and GitHub Actions, to test our code. We should add performance testing into the mix (#4201 is an old issue for this but we can open a fresh one). Currently we test every commit on every PR and we should consider if this model makes sense since performance testing will likely take longer to run than regular tests. Once developers are comfortable with their favorite tools, we can pick which ones to automate.

Make Production Data or Equivalent Available to Developers

If developers are only testing small amounts of data on their laptops, it's hard to detect performance problems. Not every bug fix requires access to data similar to production, but it should be made available. This is not a trivial task! If we are to use actual production data, we need to be very careful to de-identify it. If we start with our sample-data repo instead, we'll need to figure out how to make sure we cover cases like many files, many versions, etc.

Automate Performance Testing with Production Data or Equivalent

Hopefully the environment developers use with production data or equivalent can be made available to our CI tools. Perhaps these tests don't need to be run on every commit to every pull request, but they should be run regularly.

Use Monitoring as Performance Testing

Monitoring can be seen as a form of testing. How long is a round trip ping to production? What is the Time to First Byte? First Contentful Paint? Largest Contentful Paint? Time to Interactive? We now have a beta server that we could monitor continuously to know if our app is getting faster or slower over time. Should our monitoring of production servers be improved?

Learn from Training and Conferences

Most likely there is training available that is oriented toward performance. The subject of performance often comes up at conferences as well.

Learn from the Community How They Monitor Performance

Some members of the Dataverse community are likely users of newish tools like the ELK stack (Elasticsearch, Logstash, and Kibana), the TICK stack (Telegraph InfluxDB Chronograph and Kapacitor), GoAccess, Prometheus, Graphite, and more we haven't even heard of. In the *Monitoring* section of the Admin Guide, we already encourage the community to share findings, but we could dedicate time to this topic at our annual meeting or community calls.

Teach the Community to Do Performance Testing

We have a worldwide community of developers. We should do what we can in the form of documentation and other resources to help them develop performant code.

8.11.6 Conclusion

Given its long history, Dataverse has encountered many performance problems over the years. The core team is conversant in how to make the app more performant, but investment in learning additional tools and best practices would likely yield dividends. We should automate our performance testing, catching more problems before code is merged.

8.12 Dependency Management

Contents:

- *Introduction*
- *Terms*
- *Direct dependencies*
- *Transitive dependencies*
  - *Managing transitive dependencies in* `pom.xml`
- *Helpful tools*
- *Repositories*
- *Dataverse Parent POM*

8.12.1 Introduction

As explained under *Core Technologies*, the Dataverse Software is a Jakarta EE 8 based application that uses a lot of additional libraries for special purposes. This includes support for the SWORD API, S3 storage, and many other features.

Besides the code that glues together individual pieces, any developer needs to describe dependencies used within the Maven-based build system. As is familiar to any Maven user, this happens inside the "Project Object Model" (POM) file, `pom.xml`.

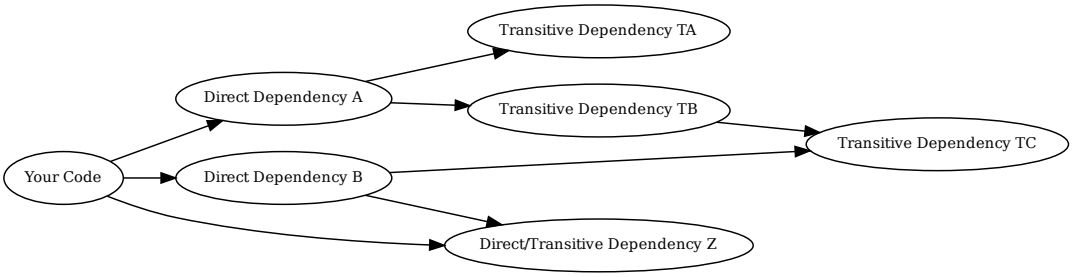
Recursive and convergent dependency resolution makes dependency management with Maven quite easy, but sometimes, in projects with many complex dependencies like the Dataverse Software, you have to help Maven make the right choices.

Maven can foster good development practices by enabling modulithic (modular monolithic) architecture: splitting functionalities into different Maven submodules while expressing dependencies between them. But there's more: the parent-child model allows you to create consistent dependency versioning (see below) within children.

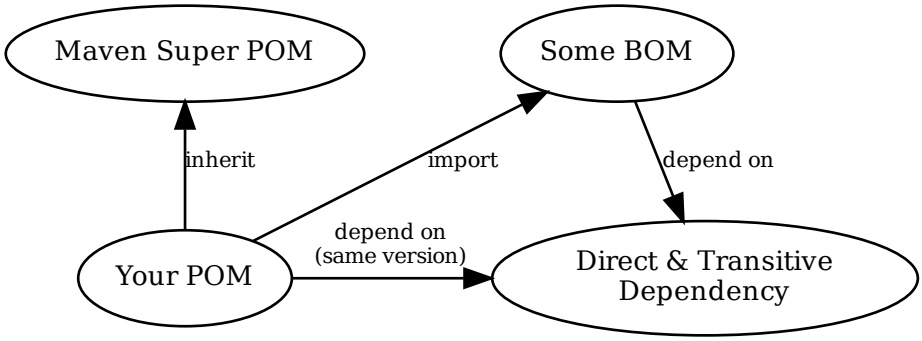
8.12.2 Terms

As a developer, you should familiarize yourself with the following terms:

- **Direct dependencies**: things *you use* yourself in your own code for the Dataverse Software.
- **Transitive dependencies**: things *others use* for things you use, pulled in recursively. See also: Maven docs.

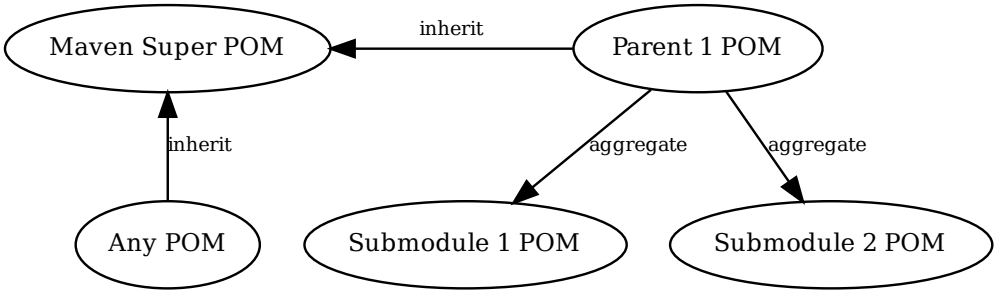


- **Project Object Model** (POM): the basic XML file unit to describe a Maven-based project.
- **Bill Of Materials** (BOM): larger projects like Payara, Amazon SDK etc. provide lists of their direct dependencies. This comes in handy when adding these dependencies (transitive for us) as direct dependencies, see below.

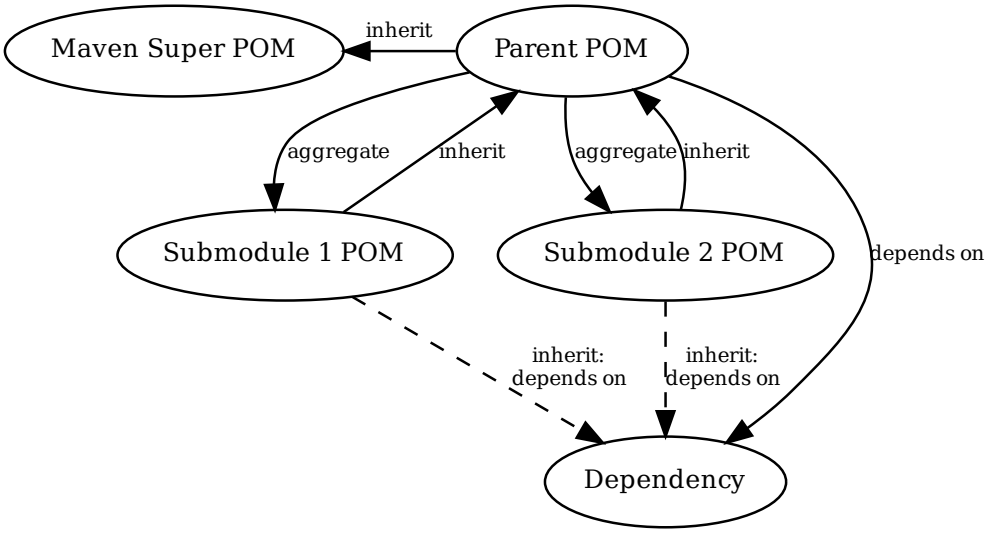


- **Parent POM**, **Super POM**: any project may be a child of a parent.

  Project silently inherit from a "super POM", which is the global Maven standard parent POM. Children may also be aggregated by a parent (without them knowing) for convenient builds of larger projects.



  Children may inherit dependencies, properties, settings, plugins etc. from the parent (making it possible to share common ground). Both approaches may be combined. Children may import as many BOMs as they want, but can have only a single parent to inherit from at a time.



- **Modules**: when using parents and children, these are called "modules" officially, each having their own POM.

  Using modules allows bundling different aspects of (Dataverse) software in their own domains, with their own behavior, dependencies etc. Parent modules allow for sharing of common settings, properties, dependencies and more. Submodules may also be used as parent modules for a lower level of submodules.

  Maven modules within the same software project may also depend on each other, allowing to create complex structures of packages and projects. Each module may be released on their own (e. g. on Maven Central) and other projects may rely on and reuse them. This is especially useful for parent POMs: they may be reused as BOMs or to share a standard between independent software projects.

  Maven modules should not be confused with the Java Platform Module System (JPMS) introduced in Java 9 under Project Jigsaw.

8.12.3 Direct dependencies

Within the POM, any direct dependencies reside within the `<dependencies>` tag:

```
<dependencies>
    <dependency>
        <groupId>org.example</groupId>
        <artifactId>example</artifactId>
        <version>1.1.0</version>
        <scope>compile</scope>
    </dependency>
</dependencies>
```

Anytime you add a `<dependency>`, Maven will try to fetch it from defined/configured repositories and use it within the build lifecycle. You have to define a `<version>` (note exception below), but `<scope>` is optional for `compile`. (See Maven docs: Dep. Scope)

During fetching, Maven will analyze all transitive dependencies (see graph above) and, if necessary, fetch those too. Everything downloaded once is cached locally by default, so nothing needs to be fetched again and again, as long as the dependency definition does not change.

Rules to follow:

1. You should only use direct dependencies for **things you are actually using** in your code.
2. When declaring a direct dependency with its **version** managed by `<dependencyManagement>`, a BOM or parent POM, you may not provide one unless you want to explicitly override!
3. **Clean up** direct dependencies no longer in use. It will bloat the deployment package otherwise!
4. Care about the **scope**[1]:
   - Do not include "testing only" dependencies in the final package - it will hurt you in IDEs and bloat things. There is scope `test` for this!
   - Make sure to use the `runtime` scope when you need to ensure a library is present on our classpath at runtime. An example is the SLF4J JUL bridge: we want to route logs from SLF4J into `java.util.logging`, so it needs to be present on the classpath, although we aren't using SLF4J unlike, some of our dependencies.
   - Some dependencies might be `provided` by the runtime environment. Good example: everything from Jakarta EE! We use the Payara BOM to ensure using the same version during development and runtime.
5. Avoid using different dependencies for the **same purpose**, e. g. different JSON parsing libraries.
6. Refactor your code to **use Jakarta EE** standards as much as possible.
7. When you rely on big SDKs or similar big cool stuff, try to **include the smallest portion possible**. Complete SDK bundles are typically heavyweight and most of the time unnecessary.
8. **Don't include transitive dependencies.**[2]
   - Exception: if you are relying on it in your code (see *Z* in the graph above), you must declare it. See below for proper handling in these (rare) cases.

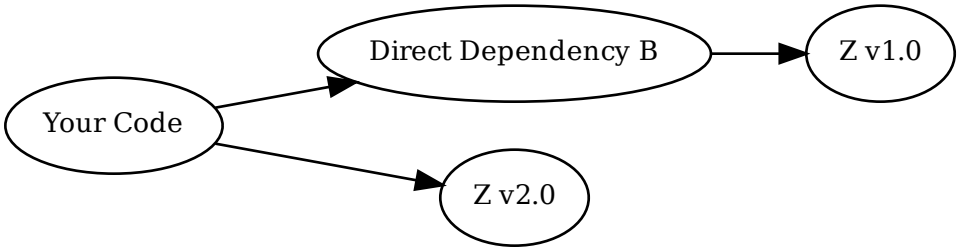
8.12.4 Transitive dependencies

Maven is comfortable for developers; it handles recursive resolution, downloading, and adding "dependencies of dependencies". However, as life is a box of chocolates, you might find yourself in *version conflict hell* sooner than later without even knowing, but experiencing unintended side effects.

When you look at the topmost graph above, imagine *B* and *TB* rely on different *versions* of *TC*. How does Maven decide which version it will include? Easy: the version of the dependency nearest to our project ("Your Code)" wins. The following graph gives an example:

[1] Modern IDEs import your Maven POM and offer import autocompletion for classes based on direct dependencies in the model. You might end up using legacy or repackaged classes because of a wrong scope.

[2] This is going to bite back in modern IDEs when importing classes from transitive dependencies by "autocompletion accident".



In this case, version "2.0" will be included. If you know something about semantic versioning, a red alert should ring in your mind right now. How do we know that *B* is compatible with *Z v2.0* when depending on *Z v1.0*?

Another scenario getting us in trouble: indirect use of transitive dependencies. Imagine the following: we rely on *Z* in our code, but do not include a direct dependency for it within the POM. Now assume *B* is updated and removed its dependency on *Z*. You definitely don't want to head down that road.

Follow the rules to be safe:

1. Do **not use transitive deps implicitly**: add a direct dependency for transitive deps you re-use in your code.
2. On every build, check that no implicit usage was added by accident.
3. **Explicitly declare versions** of transitive dependencies in use by multiple direct dependencies.
4. On every build, check that there are no convergence problems hiding in the shadows.
5. **Do special tests** on every build to verify these explicit combinations work.

Managing transitive dependencies in `pom.xml`

Maven can manage versions of transitive dependencies in four ways:

Safe Good Practice	(1) Explicitly declare the transitive dependency in `<dependencyManagement>` with a `<version>` tag.	(2) For more complex transitive dependencies, reuse a "Bill of Materials" (BOM) within `<dependencyManagement>`. Many bigger projects provide them, making the POM much less bloated compared to adding every bit yourself.
Better or Avoid Don't	(3) Use `<optional>` or `<exclusion>` tags on direct dependencies that request the transitive dependency. *Last resort*, you really should avoid this. Not explained or used here, but sometimes unavoidable. See Maven docs.	(4) Make a transitive-only dependency not used in your code a direct one and add a `<version>` tag. Typically a bad idea; don't do that.

Note: when the same transitive dependency is used in multiple Maven modules of a software project, it might be added to a common `<dependencyManagement>` section of an inherited parent POM instead. (Overrides are still possible.)

A reduced example, only showing bits relevant to the above cases and usage of an explicit transitive dep directly:

```
<properties>
    <aws.version>1.11.172</aws.version>
    <!-- We need to ensure that our choosen version is compatible with every dependency relying on it.
         This is manual work and needs testing, but a good investment in stability and up-to-date dependencies. -->
    <jackson.version>2.9.6</jackson.version>
    <joda.version>2.10.1</joda.version>
</properties>

<!-- Transitive dependencies, bigger library "bill of materials" (BOM) and
     versions of dependencies used both directly and transitive are managed here. -->
<dependencyManagement>
    <dependencies>
        <!-- First example for case 4. Only one part of the SDK (S3) is used and transitive deps
             of that are again managed by the upstream BOM. -->
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-bom</artifactId>
            <version>${aws.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
        <!-- Second example for case 4 and an example for explicit direct usage of a transitive dependency.
             Jackson is used by AWS SDK and others, but we also use it in the Dataverse Software. -->
        <dependency>
            <groupId>com.fasterxml.jackson</groupId>
            <artifactId>jackson-bom</artifactId>
            <version>${jackson.version}</version>
            <scope>import</scope>
            <type>pom</type>
        </dependency>
        <!-- Example for case 3. Joda is not used in the Dataverse Software (as of writing this). -->
        <dependency>
            <groupId>joda-time</groupId>
            <artifactId>joda-time</artifactId>
            <version>${joda.version}</version>
        </dependency>
    </dependencies>
</dependencyManagement>

<!-- Declare any DIRECT dependencies here.
     In case the depency is both transitive and direct (e. g. some common lib for logging),
     manage the version above and add the direct dependency here WITHOUT version tag, too.
-->
```

(continues on next page)

(continued from previous page)

```
<dependencies>
    <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>aws-java-sdk-s3</artifactId>
        <!-- no version here as managed by BOM above! -->
    </dependency>
    <!-- Should be refactored and removed now that we are on Jakarta EE 8 -->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-core</artifactId>
        <!-- no version here as managed above! -->
    </dependency>
    <!-- Should be refactored and removed now that we are on Jakarta EE 8 -->
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <!-- no version here as managed above! -->
    </dependency>
</dependencies>
```

8.12.5 Helpful tools

Maven provides some plugins that are of great help to detect possible conflicts and implicit usage.

For *implicit usage detection*, use `mvn dependency:analyze`. Examine the output with great care. Sometimes you will see implicit usages that do no harm, especially if you are using bigger SDKs having some kind of *core* package. This will also report on any direct dependency which is not in use and can be removed from the POM. Again, do this with great caution and double check.

If you want to see the dependencies both direct and transitive in a *dependency tree format*, use `mvn dependency:tree`.

This will however not help you with detecting possible version conflicts. For this you need to use the Enforcer Plugin with its built in dependency convergence rule.

8.12.6 Repositories

Maven receives all dependencies from *repositories*. These can be public like Maven Central and others, but you can also use a private repository on premises or in the cloud.

Repositories are defined within the Dataverse Software POM like this:

```
<repositories>
    <repository>
        <id>central-repo</id>
        <name>Central Repository</name>
        <url>http://repo1.maven.org/maven2</url>
        <layout>default</layout>
    </repository>
    <repository>
        <id>prime-repo</id>
        <name>PrimeFaces Maven Repository</name>
        <url>http://repository.primefaces.org</url>
```

(continues on next page)

(continued from previous page)

```
        <layout>default</layout>
    </repository>
</repositories>
```

You can also add repositories to your local Maven settings, see docs.

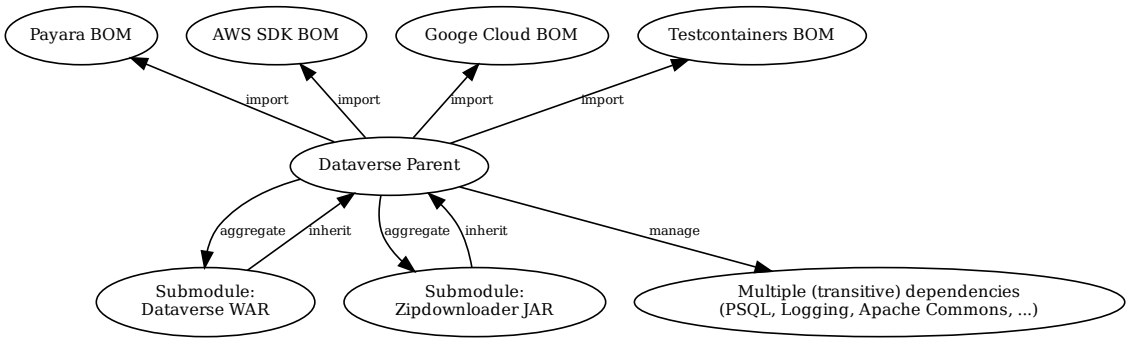
Typically you will skip the addition of the central repository, but adding it to the POM has the benefit that dependencies are first looked up there (which in theory can speed up downloads). You should keep in mind that repositories are used in the order they appear.

8.12.7 Dataverse Parent POM

Within `modules/dataverse-parent` you will find the parent POM for the Dataverse codebase. It serves different purposes:

1. Provide the common version number for a Dataverse release (may be overriden where necessary).
2. Provide common metadata necessary for releasing modules to repositories like Maven Central.
3. Declare aggregated submodules via `<modules>`.
4. Collate common BOMs and transitive dependencies within `<dependencyManagement>`. (Remember: a direct dependency declaration may omit the version element when defined in that area!)
5. Collect common `<properties>` regarding the Maven project (encoding, ...), dependency versions, target Java version, etc.
6. Gather common `<repositories>` and `<pluginRepositories>` - no need to repeat those in submodules.
7. Make submodules use current Maven plugin release versions via `<pluginManagement>`.

As of this writing (2022-02-10), our parent module looks like this:



The codebase is structured like this:

```
<project root>              # Dataverse WAR Module
├── pom.xml                 # (POM file of WAR module)
├── modules                 #
│   └── dataverse-parent    # Dataverse Parent Module
│       └── pom.xml         # (POM file of Parent Module)
└── scripts                 #
```

(continues on next page)

(continued from previous page)

```
    └── zipdownload         # Zipdownloader JAR Module
        └── pom.xml         # (POM file of Zipdownloader Module)
```

- Any developer cloning the project and running `mvn` within the project root will interact with the Dataverse WAR module, which is the same behavior since Dataverse 4.0 has been released.
- Running `mvn` targets within the parent module will execute all aggregated submodules in one go.

8.13 Debugging

Contents:

- *Logging*
- *Java Server Faces (JSF) Configuration Options*

8.13.1 Logging

By default, the app server logs at the "INFO" level but logging can be increased to "FINE" on the fly with (for example) `./asadmin set-log-levels edu.harvard.iq.dataverse.api.Datasets=FINE`. Running `./asadmin list-log-levels` will show the current logging levels.

8.13.2 Java Server Faces (JSF) Configuration Options

Some JSF options can be easily changed via MicroProfile Config (using environment variables, system properties, etc.) during development without recompiling. Changing the options will require at least a redeployment, obviously depending how you get these options in. (Variable substitution only happens during deployment and when using system properties or environment variables, you'll need to pass these into the domain, which usually will require an app server restart.)

Please note you can use MicroProfile Config to maintain your settings more easily for different environments.

JSF Option	MPCONFIG Key	Description	Default
javax.faces.PRO	dataverse.jsf.pro stage	Switch to different levels to make JSF more verbose, disable caches etc. Read more at various places.	`Production`
javax.faces.INTI _STRING_SUB	dataverse.jsf.em string-null	See Jakarta Server Faces 3.0 Spec	`true`
javax.faces.FAC	dataverse.jsf.ski comments	See Jakarta Server Faces 3.0 Spec	`true`
javax.faces.FAC	dataverse.jsf.buf size	See Jakarta Server Faces 3.0 Spec	`102400` (100 KB)
javax.faces.FAC	dataverse.jsf.refi period	See Jakarta Server Faces 3.0 Spec	`-1`
prime-faces.THEME	data-verse.jsf.primefa	See PrimeFaces Configuration Docs	`bootstrap`

8.14 Coding Style

Like all development teams, the Dataverse Project developers at IQSS have their habits and styles when it comes to writing code. Let's attempt to get on the same page. :)

- *Java*
  - *Formatting Code*
    - *Tabs vs. Spaces*
    - *Imports*
    - *Braces Placement*
    - *Format Code with Spotless*
    - *Format Code You Changed with Netbeans*
    - *Checking Your Formatting With Checkstyle*
  - *Logging*
  - *Avoid Hard-Coding Strings (Use Constants)*
  - *Avoid Hard-Coding User-Facing Messaging in English*
  - *Type Safety*
- *Bash*
  - *Formatting Code*
    - *Tabs vs. Spaces*
- *Bike Shedding*

8.14.1 Java

Formatting Code

How to format Java code is being discussed on Zulip and the dev mailing list.

Tabs vs. Spaces

Don't use tabs. Use 4 spaces.

Imports

Wildcard imports are neither encouraged nor discouraged.

Braces Placement

Place curly braces according to the style below, which is an example you can see from Netbeans.

```
public class ClassA {

    private String letters[] = new String[]{"A", "B"};

    public int meth(String text, int number) {
        BinaryOperator plus = (a, b) -> {
            return a + b;
        };
        if (text != null) {
            try {
                meth("Some text", text.length());
            } catch (Throwable t) {
            } finally {
            }
        } else if (number >= 0) {
            text = number == 0 ? "empty" : "nonempty";
        }
        do {
            number = number + 1;
        } while (number < 2);
        for (int i = 1; i < 100; i++) {
            number = number + i;
        }
        while (number > 0) {
            number--;
        }
    }
}
```

Format Code with Spotless

In some of our libraries we've had success formatting code with Spotless. See https://github.com/gdcc/xoai/issues/35 for an early discussion.

We've added Spotless to the main repo but have limited it to certain files. If you'd like to use Spotless on files you're editing, update the config in `pom.xml` to include them.

To run Spotless on your code:

```
mvn spotless:apply
```

Format Code You Changed with Netbeans

For a long time IQSS standardized on Netbeans. For files not included in the Spotless config mentioned above, it is much appreciated when you format your code (but only the code you touched) using the out-of-the-box Netbeans configuration. If you have created an entirely new Java class, you can just click Source -> Format. If you are adjusting code in an existing class, highlight the code you changed and then click Source -> Format. Keeping the "diff" in your pull requests small makes them easier to code review.

Checking Your Formatting With Checkstyle

The easiest way to adopt the Dataverse Project coding style is to use Netbeans as your IDE, avoid change the default Netbeans formatting settings, and only reformat code you've changed, as described above.

If you do not use Netbeans, you are encouraged to check the formatting of your code using Checkstyle.

To check the entire project:

```
mvn checkstyle:checkstyle
```

To check a single file:

```
mvn checkstyle:checkstyle -Dcheckstyle.includes=**\/SystemConfig*.java
```

Logging

We have adopted a pattern where the top of every class file has a line like this:

```
private static final Logger logger = Logger.getLogger(DatasetUtil.class.
↪getCanonicalName());
```

Use this `logger` field with varying levels such as `fine` or `info` like this:

```
logger.fine("will get thumbnail from dataset logo");
```

Generally speaking you should use `fine` for everything that you don't want to show up by default in the app server's log file. If you use a higher level such as `info` for common operations, you will probably hear complaints that your code is too "chatty" in the logs. These logging levels can be controlled at runtime both on your development machine and in production as explained in the *Debugging* section.

When adding logging, do not simply add `System.out.println()` lines because the logging level cannot be controlled.

Avoid Hard-Coding Strings (Use Constants)

Special strings should be defined as public constants. For example, `DatasetFieldConstant.java` contains a field for "title" and it's used in many places in the code (try "Find Usages" in Netbeans). This is better than writing the string "title" in all those places.

Avoid Hard-Coding User-Facing Messaging in English

There is an ongoing effort to translate the Dataverse Software into various languages. Look for "lang" or "languages" in the *Configuration* section of the Installation Guide for details if you'd like to help or play around with this feature.

The translation effort is hampered if you hard code user-facing messages in English in the Java code. Put English strings in `Bundle.properties` and use `BundleUtil` to pull them out. This is especially important for messages that appear in the UI. We are aware that the API has many, many hard coded English strings in it. If you touch a method in the API and notice English strings, you are strongly encouraged to use that opportunity to move the English to `Bundle.properties`.

Type Safety

If you just downloaded Netbeans and are using the out-of-the-box settings, you should be in pretty good shape. Unfortunately, the default configuration of Netbeans doesn't warn you about type-safety problems you may be inadvertently introducing into the code. To see these warnings, click Netbeans -> Preferences -> Editor -> Hints and check the following:

- "Raw Types" under "Standard Javac Warnings"

If you know of a way to easily share Netbeans configuration across a team, please get in touch.

8.14.2 Bash

Generally, Google's Shell Style Guide at https://google.github.io/styleguide/shell.xml seems to have good advice.

Formatting Code

Tabs vs. Spaces

Don't use tabs. Use 2 spaces.

`shfmt` from https://github.com/mvdan/sh seems like a decent way to enforce indentation of two spaces (i.e. `shfmt -i 2 -w path/to/script.sh`) but be aware that it makes other changes.

8.14.3 Bike Shedding

What color should the bike shed be? :)

Come debate with us about coding style in this Google doc that has public comments enabled: https://docs.google.com/document/d/1KTd3FpM1BI3HlBofaZjMmBiQEJtFf11jiGpQeJzy7A/edit?usp=sharing

8.15 Consuming Configuration

Contents:

- *Simple Configuration Options*
- *Complex Configuration Options*
- *Why should I care about MicroProfile Config API?*
- *Adopting MicroProfile Config API*
  - *Adding a JVM Setting*
  - *Moving or Replacing a JVM Setting*
  - *Adding a Feature Flag*

The Dataverse Software uses different types of configuration:

1. JVM system properties
2. Simple database value settings
3. Complex database stored data structures

1 and 2 are usually simple text strings, boolean switches or digits. All of those can be found in *Configuration*.

Anything for 3 is configured via the API using either TSV or JSON structures. Examples are metadata blocks, authentication providers, harvesters and others.

8.15.1 Simple Configuration Options

Developers can access simple properties via:

1. `JvmSettings.<SETTING NAME>.lookup(...)` for JVM system property settings.
2. `SettingsServiceBean.get(...)` for database settings.
3. `SystemConfig.xxx()` for specially treated settings, maybe mixed from 1 and 2 and other sources.
4. `SettingsWrapper` for use in frontend JSF (xhtml) pages to obtain settings from 2 and 3. Using the wrapper is a must for performance as explained in *avoid common efficiency issues with JSF render logic expressions*.
5. `System.getProperty()` only for very special use cases not covered by `JvmSettings`.

As of Dataverse Software 5.3, we start to streamline our efforts into using a more consistent approach, also bringing joy and happiness to all the system administrators out there. This will be done by adopting the use of MicroProfile Config over time.

So far we streamlined configuration of these Dataverse Software parts:

- Database Connection

8.15.2 Complex Configuration Options

We should enable variable substitution in JSON configuration. Example: using substitution to retrieve values from MicroProfile Config and insert into the authentication provider would allow much easier provisioning of secrets into the providers.

8.15.3 Why should I care about MicroProfile Config API?

Developers benefit from:

- A streamlined API to retrieve configuration, backward-compatible renaming strategies and easier testbed configurations.
- Config API is also pushing for validation of configuration, as it's typesafe and converters for non-standard types can be added within our codebase.
- Defaults in code or bundled in `META-INF/microprofile-config.properties` allow for optional values without much hassle.
- A single place to lookup any existing JVM setting in code, easier to keep in sync with the documentation.

System administrators benefit from:

- Lots of database settings have been introduced in the past, but should be more easily configurable and not rely on a database connection.
- Running a Dataverse installation in containers gets much easier when configuration can be provisioned in a streamlined fashion, mitigating the need for scripting glue and distinguishing between setting types.
- Classic installations have a profit, too: we can enable using a single config file, e.g. living in `/etc/dataverse/config.properties` by adding our own, hot-reload config source.
- Features for monitoring resources and others are easier to use with this streamlined configuration, as we can avoid people having to deal with `asadmin` commands and change a setting with comfort instead.

8.15.4 Adopting MicroProfile Config API

This technology is introduced on a step-by-step basis. There will not be a big shot, crashing upgrades for everyone. Instead, we will provide backward compatibility by deprecating renamed or moved config options, while still supporting the old way of setting them.

- Introducing a new setting or moving an old one should result in a scoped key `dataverse.<scope/task/module/...>.<setting>`. That way we enable sys admins to recognize the meaning of an option and avoid name conflicts. Starting with `dataverse` makes it perfectly clear that this is a setting meant for this application, which is important when using environment variables, system properties or other MPCONFIG sources.
- Replace `System.getProperty()` calls with `JvmSettings.<SETTING NAME>.lookup(...)`, adding the setting there first. This might be paired with renaming and providing backward-compatible aliases.
- Database settings need to be refactored in multiple steps and it is not yet clear how this will be done. Many Database settings are of very static nature and might be moved to JVM settings (in backward compatible ways).

Adding a JVM Setting

Whenever a new option gets added or an existing configuration gets migrated to `edu.harvard.iq.dataverse.settings.JvmSettings`, you will attach the setting to an existing scope or create new sub-scopes first.

- Scopes and settings are organised in a tree-like structure within a single enum `JvmSettings`.
- The root scope is "dataverse".
- All sub-scopes are below that.
- Scopes are separated by dots (periods).
- A scope may be a placeholder, filled with a variable during lookup. (Named object mapping.)
- The setting should be in kebab case (`signing-secret`) rather than camel case (`signingSecret`).

Any consumer of the setting can choose to use one of the fluent `lookup()` methods, which hides away alias handling, conversion etc from consuming code. See also the detailed Javadoc for these methods.

Moving or Replacing a JVM Setting

When moving an old key to a new (especially when doing so with a former JVM system property setting), you should add an alias to the `JvmSettings` definition to enable backward compatibility. Old names given there are capable of being used with patterned lookups.

Another option is to add the alias in `src/main/resources/META-INF/microprofile-aliases.properties`. The format is always like `dataverse.<scope/....>.newname...=old.property.name`. Note this doesn't provide support for patterned aliases.

Details can be found in `edu.harvard.iq.dataverse.settings.source.AliasConfigSource`

Adding a Feature Flag

Some parts of our codebase might be opt-in only. Experimental or optional feature previews can be switched on using our usual configuration mechanism, a JVM setting.

Feature flags are implemented in the enumeration `edu.harvard.iq.dataverse.settings.FeatureFlags`, which allows for convenient usage of it anywhere in the codebase. When adding a flag, please add it to the enum, think of a default status, add some Javadocs about the flagged feature and add a `@since` tag to make it easier to identify when a flag has been introduced.

Add the feature flag to the list at *feature flags*.

8.16 Deployment

Developers often only deploy the Dataverse Software to their *Development Environment* but it can be useful to deploy the Dataverse Software to cloud services such as Amazon Web Services (AWS).

Contents:

- *Deploying the Dataverse Software to Amazon Web Services (AWS)*
  - *Install AWS CLI*

        * *Troubleshooting "aws: command not found"*
    - *Configure AWS CLI with Stored Credentials*
    - *Configure Role-Based S3 Access*
    - *Configure Ansible File (Optional)*
    - *Download and Run the "Create Instance" Script*
    - *Caveat Recipiens*
    - *Migrating Datafiles from Local Storage to S3*
        * *To Update Dataset Location to S3, Assuming a `file://` Prefix*
        * *To Update Datafile Location to your-s3-bucket, Assuming a `file://` Prefix*
        * *To Update Datafile Location to your-s3-bucket, Assuming no `file://` Prefix*

8.16.1 Deploying the Dataverse Software to Amazon Web Services (AWS)

We have written scripts to deploy the Dataverse Software to Amazon Web Services (AWS) but they require some setup.

Install AWS CLI

First, you need to have AWS Command Line Interface (AWS CLI) installed, which is called `aws` in your terminal. Launching your terminal and running the following command to print out the version of AWS CLI will tell you if it is installed or not:

```
aws --version
```

If you have not yet installed AWS CLI you should install it by following the instructions at https://docs.aws.amazon.com/cli/latest/userguide/installing.html

Afterwards, you should re-run the "version" command above to verify that AWS CLI has been properly installed. If "version" still doesn't work, read on for troubleshooting advice. If "version" works, you can skip down to the "Configure AWS CLI" step.

Troubleshooting "aws: command not found"

Please note that as of this writing the AWS docs are not especially clear about how to fix errors such as `aws: command not found`. If the AWS CLI cannot be found after you followed the AWS installation docs, it is very likely that the `aws` program is not in your `$PATH`. `$PATH` is an "environment variable" that tells your shell (usually Bash) where to look for programs.

To see what `$PATH` is set to, run the following command:

```
echo $PATH
```

On Mac, to update your `$PATH` to include the location where the current AWS docs install AWS CLI on the version of Python included with your Mac, run the following command:

```
export PATH=$PATH:$HOME/Library/Python/2.7/bin
```

After all this, you can try the "version" command again.

Note that it's possible to add an `export` line like the one above to your `~/.bash_profile` file so you don't have to run it yourself when you open a new terminal.

Configure AWS CLI with Stored Credentials

Dataverse can access S3 using credentials stored as described below, or using an IAM role described a little further below.

Create a `.aws` directory in your home directory (which is called ~) like this:

```
mkdir ~/.aws
```

We will be creating two plain text files in the `.aws` directory and it is important that these files do not end in ".txt" or any other extension. After creating the files, you can verify their names with the following command:

```
ls ~/.aws
```

Create a plain text file at `~/.aws/config` with the following content:

```
[default]
region = us-east-1
```

Please note that at this time the region must be set to "us-east-1" but in the future we could improve our scripts to support other regions.

Create a plain text file at `~/.aws/credentials` with the following content:

```
[default]
aws_access_key_id = XXXXXXXXXXXXXXXXXXX
aws_secret_access_key = XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Then update the file and replace the values for "aws_access_key_id" and "aws_secret_access_key" with your actual credentials by following the instructions at https://aws.amazon.com/blogs/security/wheres-my-secret-access-key/

If you are having trouble configuring the files manually as described above, see https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html which documents the `aws configure` command.

Configure Role-Based S3 Access

Amazon offers instructions on using an IAM role to grant permissions to applications running in EC2 at https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html

Configure Ansible File (Optional)

In order to configure Dataverse installation settings such as the password of the dataverseAdmin user, download https://raw.githubusercontent.com/gdcc/dataverse-ansible/develop/defaults/main.yml and edit the file to your liking.

You can skip this step if you're fine with the values in the "main.yml" file in the link above.

Download and Run the "Create Instance" Script

Once you have done the configuration above, you are ready to try running the "ec2-create-instance.sh" script to spin up a Dataverse installation in AWS.

Download ec2-create-instance.sh and put it somewhere reasonable. For the purpose of these instructions we'll assume it's in the "Downloads" directory in your home directory.

To run the script, you can make it executable (`chmod 755 ec2-create-instance.sh`) or run it with bash, like this with `-h` as an argument to print the help:

```
bash ~/Downloads/ec2-create-instance.sh -h
```

If you run the script without any arguments, it should spin up the latest version of Dataverse.

You will need to wait for 15 minutes or so until the deployment is finished, longer if you've enabled sample data and/or the API test suite. Eventually, the output should tell you how to access the Dataverse installation in a web browser or via SSH. It will also provide instructions on how to delete the instance when you are finished with it. Please be aware that AWS charges per minute for a running instance. You may also delete your instance from https://console.aws.amazon.com/console/home?region=us-east-1 .

Caveat Recipiens

Please note that while the script should work well on new-ish branches, older branches that have different dependencies such as an older version of Solr may not produce a working Dataverse installation. Your mileage may vary.

Migrating Datafiles from Local Storage to S3

A number of pilot Dataverse installations start on local storage, then administrators are tasked with migrating datafiles into S3 or similar object stores. The files may be copied with a command-line utility such as s3cmd. You will want to retain the local file hierarchy, keeping the authority (for example: 10.5072) at the bucket "root."

The below example queries may assist with updating dataset and datafile locations in the Dataverse installation's PostgresQL database. Depending on the initial version of the Dataverse Software and subsequent upgrade path, Datafile storage identifiers may or may not include a `file://` prefix, so you'll want to catch both cases.

In the following queries, `<id>://` refers to the `id` you have set for the datastore in the configuration (see the *File Storage* section of the Installation Guide).

To Update Dataset Location to S3, Assuming a `file://` Prefix

```
UPDATE dvobject SET storageidentifier=REPLACE(storageidentifier,'file://','<id>://')
  WHERE dtype='Dataset';
```

To Update Datafile Location to your-s3-bucket, Assuming a `file://` Prefix

```
UPDATE dvobject
  SET storageidentifier=REPLACE(storageidentifier,'file://','<id>://your-s3-bucket:')
  WHERE id IN (SELECT o.id FROM dvobject o, dataset s WHERE o.dtype = 'DataFile'
  AND s.id = o.owner_id AND s.harvestingclient_id IS null
  AND o.storageidentifier NOT LIKE '<id>://%');
```

To Update Datafile Location to your-s3-bucket, Assuming no `file://` Prefix

```
UPDATE dvobject SET storageidentifier=CONCAT('<id>://your-s3-bucket:', storageidentifier)
        WHERE id IN (SELECT o.id FROM dvobject o, dataset s WHERE o.dtype = 'DataFile'
        AND s.id = o.owner_id AND s.harvestingclient_id IS null
        AND o.storageidentifier NOT LIKE '%://%');
```

8.17 Docker, Kubernetes, and Containers

The Dataverse community is exploring the use of Docker, Kubernetes, and other container-related technologies.

Contents:

- *Container Guide*
- *Help Containerize Dataverse*
- *Community-Lead Projects*
- *Using Containers for Reproducible Research*

8.17.1 Container Guide

We recommend starting with the *Container Guide*. The core Dataverse development team, with lots of help from the community, is iterating on containerizing the Dataverse software and its dependencies there.

8.17.2 Help Containerize Dataverse

If you would like to contribute to the containerization effort, please consider joining the Containerization Working Group.

8.17.3 Community-Lead Projects

The primary community-lead projects (which the core team is drawing inspiration from!) are:

- https://github.com/IQSS/dataverse-docker
- https://github.com/IQSS/dataverse-kubernetes (especially the https://github.com/EOSC-synergy/dataverse-kubernetes fork)

8.17.4 Using Containers for Reproducible Research

Please see *Research Code* in the User Guide for this related topic.

8.18 Making Releases

Contents:

- *Introduction*
- *Regular or Hotfix?*
- *Ensure Issues Have Been Created*
- *Declare a Code Freeze*
- *Push Back Milestones on Pull Requests That Missed the Train*
- *Write Release Notes*
- *Build Release Candidate*
- *Deploy Release Candidate to Internal*
- *Deploy Release Candidate to QA*
- *Solicit Feedback from Curation Team*
- *Conduct Performance Testing*
- *Conduct Regression Testing*
- *Build the Guides for the Release Candidate*
- *Deploy Release Candidate to Demo*
- *Merge Release Notes (Once Ready)*
- *Prepare Release Branch*
- *Merge "develop" into "master" (non-hotfix only)*
- *Confirm "master" Mergeability*
- *Add Milestone to Pull Requests and Issues*
- *Build the Guides for the Release*
- *Run a Build to Create the War File*
- *Build Installer (dvinstall.zip)*

- *Create a Draft Release on GitHub*
- *Publish the Release*
- *Update Guides Link*
- *Test Docker Images*
- *Close Milestone on GitHub and Create a New One*
- *Update the Container Base Image Version Property*
- *Deploy Final Release on Demo*
- *Update SchemaSpy*
- *Alert Translators About the New Release*
- *Add the Release to the Dataverse Roadmap*
- *Announce the Release on the Dataverse Blog*
- *Announce the Release on the Mailing List*
- *Announce the Release on Zulip*
- *For Hotfixes, Merge Hotfix Branch into "develop"*
- *For Hotfixes, Tell Developers to Merge "develop" into Their Branches and Rename SQL Scripts*
- *Lift the Code Freeze and Encourage Developers to Update Their Branches*

8.18.1 Introduction

Note

This document is about releasing the main Dataverse app (https://github.com/IQSS/dataverse). See *Making Library Releases* for how to release our various libraries. Other projects have their own release documentation.

Note

Below you'll see branches like "develop" and "master" mentioned. For more on our branching strategy, see *Version Control*.

Dataverse releases are time-based as opposed to being feature-based. That is, we announce an approximate release date in advance (e.g. for 6.8) and try to hit that deadline. If features we're working on aren't ready yet, the train will leave the station without them. We release quarterly.

We also announce "last call" dates for both community pull requests and those made by core developers. If you are part of the community and have made a pull request, you have until this date to ask the team to add the upcoming milestone to your pull request. The same goes for core developers. This is not a guarantee that these pull requests will be reviewed, tested, QA'ed and merged before *code freeze*, but we'll try.

8.18.2 Regular or Hotfix?

Early on, make sure it's clear what type of release this is. The steps below describe making both regular releases and hotfix releases.

- regular
  - e.g. 6.5 (minor)
  - e.g. 7.0 (major)
- hotfix
  - e.g. 6.4.1 (patch)
  - e.g. 7.0.1 (patch)

8.18.3 Ensure Issues Have Been Created

We have a "create release issues" script at https://github.com/IQSS/dv-project-metrics that should be run a week or so before code freeze.

For each issue that is created by the script there is likely a corresponding step in this document that has "dedicated" label on it like this:

There are a variety of reasons why a step might deserve its own dedicated issue:

- The step can be done by a team member other than the person doing the release.
- Stakeholders might be interested in the status of a step (e.g. has the release been deployed to the demo site).

Steps don't get their own dedicated issue if it would be confusing to have multiple people involved. Too many cooks in the kitchen, as they say. Also, some steps are so small the overhead of an issue isn't worth it.

8.18.4 Declare a Code Freeze

When we declare a code freeze, we mean:

- No additional features will be merged until the freeze is lifted.
- Bug fixes will only be merged if they relate to the upcoming release in some way, such as fixes for regressions or performance problems in that release.
- Pull requests that directly affect the release, such as bumping the version, will be merged, of course.

The benefits of the code freeze are:

- The team can focus on getting the release out together.
- Regression and performance testing can happen on code that isn't changing.
- The release notes can be written without having to worry about new features (and their release note snippets) being merged in.

In short, the steps described below become easier under a code freeze.

Note: for a hotfix, a code freeze is necessary not because we want code to stop changing in the branch being hotfix released, but because bumping the version used in Jenkins/Ansible means that API tests will fail in pull requests until the version is bumped in those pull requests. Basically, we want to get the hotfix merged quickly so we can propagate the version bump into all open pull requests so that API tests can start passing again in those pull requests.

8.18.5 Push Back Milestones on Pull Requests That Missed the Train

As of this writing, we optimistically add milestones to issues and pull requests, hoping that the work will be complete before code freeze. Inevitably, we're a bit too optimistic.

Hopefully, as the release approached, the team has already decided which pull requests (that aren't related to the release) won't make the cut. If not, go ahead and bump them to the next release.

8.18.6 Write Release Notes

Developers express the need for an addition to release notes by creating a "release note snippet" in `/doc/release-notes` containing the name of the issue they're working on. The name of the branch could be used for the filename with ".md" appended (release notes are written in Markdown) such as `5053-apis-custom-homepage.md`. See *Writing a Release Note Snippet* for how this is described for contributors.

The task at or near release time is to collect these snippets into a single file.

- Find the issue in GitHub that tracks the work of creating release notes for the upcoming release.
- Create a branch, add a .md file for the release (ex. 6.10.1 Release Notes) in `/doc/release-notes` and write the release notes, making sure to pull content from the release note snippets mentioned above. Snippets may not include any issue number or pull request number in the text so be sure to copy the number from the filename of the snippet into the final release note.
- Delete (`git rm`) the release note snippets as the content is added to the main release notes file.
- Include instructions describing the steps required to upgrade the application from the previous version. These must be customized for release numbers and special circumstances such as changes to metadata blocks and infrastructure. These instructions are required for the next steps (deploying to various environments) so try to prioritize them over finding just the right words in release highlights (which you can do later).
- We usually include a "Security Updates" section due to dependencies we've updated. Under that section, give credit to any security researchers who have reported vulnerabilities. In the 6.11 release notes, for example, we wrote "We would like to thank [person1], [person2], and [person3] for notifying us about vulnerabilities that were fixed in this release." See *Giving Credit to Security Researchers* and *Reporting Security Issues*.
- Make a pull request. Here's an example: https://github.com/IQSS/dataverse/pull/11613
- Note that we won't merge the release notes until after we have confirmed that the upgrade instructions are valid by performing a couple upgrades.

For a hotfix, don't worry about release notes yet.

8.18.7 Build Release Candidate

Go to https://github.com/IQSS/dataverse/actions/workflows/generate_war_file.yml click "run workflow". For a regular release, make sure the branch is "develop". For a hotfix, you will use whatever branch name is used for the hotfix. Leave the custom label blank and click "run workflow". This will create an action that should result in a zip file. Inside that zip is another zip that contains the war file.

8.18.8 Deploy Release Candidate to Internal

ssh into the dataverse-internal server and download the release candidate war file you built above.

Go to /doc/release-notes, open the release-notes.md file for the release we're working on, and perform all the steps under "Upgrade Instructions". Note that for regular releases, we haven't bumped the version yet so you won't be able to follow the steps exactly. (For hotfix releases, the version will be bumped already.)

8.18.9 Deploy Release Candidate to QA

Deploy the same war file to https://qa.dataverse.org using the same upgrade instructions as above.

8.18.10 Solicit Feedback from Curation Team

Ask the curation team to test on https://qa.dataverse.org and give them five days to provide feedback.

8.18.11 Conduct Performance Testing

See *Locust* and <https://github.com/IQSS/dataverse/issues/12284>, for example.

8.18.12 Conduct Regression Testing

Regression testing should be conducted on production data.

8.18.13 Build the Guides for the Release Candidate

Go to https://jenkins.dataverse.org/job/guides.dataverse.org/ and make the following adjustments to the config:

- Repository URL: `https://github.com/IQSS/dataverse.git`
- Branch Specifier (blank for 'any'): `*/develop`
- `VERSION` (under "Build Steps"): use the next release version but add "-rc.1" to the end. Don't prepend a "v". Use `6.8-rc.1` (for example)

Click "Save" then "Build Now".

Make sure the guides directory appears in the expected location such as https://guides.dataverse.org/en/6.8-rc.1/

When previewing the HTML version of docs from pull requests, we don't usually use this Jenkins job, relying instead on automated ReadTheDocs builds. The reason for doing this step now while we wait for feedback from the Curation Team is that it's an excellent time to fix the Jenkins job, if necessary, to accommodate any changes needed to continue to build the docs. For example, Sphinx might need to be updated or a dependency might need to be installed. Such changes should be listed in the release notes for documentation writers.

8.18.14 Deploy Release Candidate to Demo

Time has passed. The curation team has given feedback. We've finished regression and performance testing. Fixes may have been merged into the "develop" branch. We're ready to actually make the release now, which includes deploying a release candidate to the demo server.

Build a new war file, if necessary, and deploy it to https://demo.dataverse.org using the upgrade instructions in the release notes.

8.18.15 Merge Release Notes (Once Ready)

If the upgrade instructions are perfect, simply merge the release notes.

If the upgrade instructions aren't quite right, work with the authors of the release notes until they are good enough, and then merge.

For a hotfix, there are no release notes to merge yet.

8.18.16 Prepare Release Branch

The release branch will have the final changes such as bumping the version number.

Usually we branch from the "develop" branch to create the release branch. If we are creating a hotfix for a particular version (5.11, for example), we branch from the tag (e.g. `v5.11`).

Create a release branch named after the issue that tracks bumping the version with a descriptive name like "10852-bump-to-6.4" from https://github.com/IQSS/dataverse/pull/10871.

Note: the changes below must be the very last commits merged into the develop branch before it is merged into master and tagged for the release!

Make the following changes in the release branch.

Increment the version number to the milestone (e.g. 6.10.1) in the following two files:

- modules/dataverse-parent/pom.xml -> <properties> -> <revision> (e.g. pom.xml commit)
- doc/sphinx-guides/source/conf.py

In the following `versions.rst` file:

- doc/sphinx-guides/source/versions.rst - Below the `- |version|` bullet (`|version|` comes from the `conf.py` file you just edited), add a bullet for what is soon to be the previous release.

Return to the parent pom and make the following change, which is necessary for proper tagging of images:

- modules/dataverse-parent/pom.xml -> <profiles> -> profile "ct" -> <properties> -> Set <base.image.version> to `${revision}`

When testing the version change in Docker note that you will have to build the base image manually. See *Build Instructions*.

(Before you make this change the value should be `${parsedVersion.majorVersion}.${parsedVersion.nextMinorVersion}`. Later on, after cutting a release, we'll change it back to that value.)

For a regular release, make the changes above in the release branch you created, but hold off for a moment on making a pull request because Jenkins will fail because it will be testing the previous release.

In the dataverse-ansible repo bump the version in jenkins.yml and make a pull request such as https://github.com/gdcc/dataverse-ansible/pull/386. Wait for it to be merged. Note that bumping on the Jenkins side like this will mean that all pull requests will show failures in Jenkins until they are updated to the version we are releasing.

Once dataverse-ansible has been merged, return to the branch you created above ("10852-bump-to-6.4" or whatever) and make a pull request. Ensure that all tests are passing and then put the PR through the normal review and QA process.

If you are making a hotfix release, <base.image.version> should already be set to ${revision}. If so, leave it alone. Go ahead and do the normal bumping of version numbers described above. Make the pull request against the "master" branch. Put it through review and QA. Do not delete the branch after merging because we will later merge it into the "develop" branch to pick up the hotfix. More on this later.

8.18.17 Merge "develop" into "master" (non-hotfix only)

If this is a regular (non-hotfix) release, create a pull request to merge the "develop" branch into the "master" branch using this "compare" link: https://github.com/IQSS/dataverse/compare/master...develop

Allow time for important tests pass:

- Unit tests: Maven Tests
- API tests: Container Integration Tests Workflow

Don't worry about style tests failing such as for shell scripts.

It's ok to skip code review.

When merging the pull request, be sure to choose "create a merge commit" and not "squash and merge" or "rebase and merge". We suspect that choosing squash or rebase may have led to lots of merge conflicts when we tried to perform this "merge develop to master" step, forcing us to re-do the previous release before we could proceed with the current release.

If this is a hotfix release, skip this whole "merge develop to master" step (the "develop" branch is not involved until later).

8.18.18 Confirm "master" Mergeability

Hopefully, the previous step went ok. As a sanity check, use the "compare" link at https://github.com/IQSS/dataverse/compare/master...develop again to look for merge conflicts without making a pull request.

If the GitHub UI tells you there would be merge conflicts, something has gone horribly wrong (again) with the "merge develop to master" step. Stop and ask for help.

8.18.19 Add Milestone to Pull Requests and Issues

Often someone is making sure that the proper milestone (e.g. 6.10.1) is being applied to pull requests and issues, but sometimes this falls between the cracks.

Check for merged pull requests that have no milestone by going to https://github.com/IQSS/dataverse/pulls and entering is:pr is:merged no:milestone as a query. If you find any, first check if those pull requests are against open pull requests. If so, do nothing. Otherwise, add the milestone to the pull request and any issues it closes. This includes the "merge develop into master" pull request above.

8.18.20 Build the Guides for the Release

Go to https://jenkins.dataverse.org/job/guides.dataverse.org/ and make the following adjustments to the config:

- Repository URL: `https://github.com/IQSS/dataverse.git`
- Branch Specifier (blank for 'any'): `*/master`
- `VERSION` (under "Build Steps"): bump to the next release. Don't prepend a "v". Use `6.10.1` (for example)

Click "Save" then "Build Now".

Make sure the guides directory appears in the expected location such as https://guides.dataverse.org/en/6.10.1/

As described below, we'll soon point the "latest" symlink to that new directory.

8.18.21 Run a Build to Create the War File

Go to https://github.com/IQSS/dataverse/actions/workflows/generate_war_file.yml click "run workflow". For a regular release, change the branch to "master". For a hotfix release, use whatever branch name is used for the hotfix. Leave the custom label blank and click "run workflow". This will create an action that should result in a zip file. Inside that zip is another zip that contains the war file. Download it.

The build number will appear in `/api/info/version` (along with the commit mentioned above) from a running installation (e.g. `{"version":"6.10.1","build":"master-300d5b5"}`).

8.18.22 Build Installer (dvinstall.zip)

In a git checkout of the source, switch to the master branch and pull the latest.

Unzip the zip file from the previous step.

Copy the war file to the `target` directory in the root of the repo (create the `target` directory, if necessary):

```
cp ~/Downloads/built-app.zip .
unzip built-app.zip
rm built-app.zip
mkdir -p target
mv dataverse-*.war target
```

Then, create the installer:

```
cd scripts/installer
make clean
make
```

A zip file called `dvinstall.zip` should be produced.

8.18.23 Create a Draft Release on GitHub

Go to https://github.com/IQSS/dataverse/releases/new to start creating a draft release.

- Under "Select tag" you will be creating a new tag. Have it start with a "v" such as `v6.10.1`. Click "Create new tag". Don't worry, the tag won't be created until you publish.
- Under "Target", choose "master". This commit will appear in `/api/info/version` from a running installation.
- Under "Release title" use the same name as the tag such as `v6.10.1`.
- In the description, copy and paste the content from the release notes .md file created in the "Write Release Notes" steps above.
- Under "attach binaries", upload the war file and installer you created above.
- Click "Save draft" because we do not want to publish the release yet.

At this point you can send around the draft release for any final feedback. Links to the guides for this release should be working now, since you build them above.

Make corrections to the draft, if necessary. It will be out of sync with the .md file, but that's ok (#7988 is tracking this).

8.18.24 Publish the Release

Click the "Publish release" button.

8.18.25 Update Guides Link

"latest" at https://guides.dataverse.org/en/latest/ is a symlink to the directory with the latest release. That directory (e.g. `6.10.1`) was put into place by the Jenkins "guides" job described above.

ssh into the guides server and update the symlink to point to the latest release, as in the example below.

```
cd /var/www/html/en
ln -s 6.10.1 latest
```

This step could be done before publishing the release if you'd like to double check that links in the release notes work.

8.18.26 Test Docker Images

Publishing the release should have trigged the "Container Images Scheduled Maintenance" GitHub Action. Allow it to finish and then go to https://hub.docker.com/u/gdcc and navigate to "gdcc/dataverse".

Click on "tags" and look at the "latest" tag. Was it just updated? Good! If not, we plan to address this is https://github.com/IQSS/dataverse/issues/12514 but for now, as a workaround, run the action again. Go to https://github.com/IQSS/dataverse/actions/workflows/container_maintenance.yml and click the "run workflow" dropdown. Make sure the branch is set to "develop" and click "run workflow" button.

Wait for the action to finish and then check again that the "latest" tag has been updated.

Locally, delete old images and spin up the "latest" tag.

```
docker rmi gdcc/dataverse:latest
docker rmi gdcc/configbaker:latest
cd docker/compose/demo
```

(continues on next page)

(continued from previous page)

```
rm -rf data
docker compose up
```

Wait for the bootstrapping process to complete. Then, look at http://localhost:8080/api/info/version to make sure "version" shows the version that you just released. Note that it's normal for "build" to be null for our Docker images.

8.18.27 Close Milestone on GitHub and Create a New One

You can find our milestones at https://github.com/IQSS/dataverse/milestones

Now that we've published the release, close the milestone and create a new one for the **next** release, the release **after** the one we're working on, that is.

Note that for milestones we use just the number without the "v" (e.g. "6.10.1").

On the project board at https://github.com/orgs/IQSS/projects/34 edit the tab (view) that shows the milestone to show the next milestone.

8.18.28 Update the Container Base Image Version Property

Create a new branch (any name is fine but `prepare-next-iteration` is suggested) and update the following files to prepare for the next development cycle:

- modules/dataverse-parent/pom.xml -> <profiles> -> profile "ct" -> <properties> -> Set <base.image.version> to ${parsedVersion.majorVersion}.${parsedVersion.nextMinorVersion}

Create a pull request.

Wait for checks to complete. It's ok (even expected) for the following check to fail:

- main-integration-tests-workflow from container_integration_tests.yml
  - If you see an error like `Error: DOCKER> Unable to pull 'gdcc/base:6.12-noble-p7.2026.2-j21' : {"message":"manifest for gdcc/base:6.12-noble-p7.2026.2-j21 not found: manifest unknown: manifest unknown"} (Not Found: 404) [{"message":"manifest for gdcc/base:6.12-noble-p7.2026.2-j21 not found: manifest unknown: manifest unknown"} (Not Found: 404)]` it's telling you that the Docker image can't be spun up for API testing because it doesn't exist yet. (The error above was just after the 6.11 release.) The image will exist once the pull request is approved and merged.

Put the pull request through code review, like usual, but make sure reviewers know it's ok to ignore the check above.

Give it a milestone of the next release, the one **after** the one we're working on. Once the pull request has been approved, merge it. It should be the first PR merged of the next release.

For more background, see *Supported Image Tags*. For an example, see https://github.com/IQSS/dataverse/pull/10896

For a hotfix, we will do this later and in a different branch. See below.

8.18.29 Deploy Final Release on Demo

Above you already did the hard work of deploying a release candidate to https://demo.dataverse.org. It should be relatively straightforward to undeploy the release candidate and deploy the final release.

8.18.30 Update SchemaSpy

We maintain SchemaSpy at URLs like https://guides.dataverse.org/en/latest/schemaspy/index.html and (for example) https://guides.dataverse.org/en/6.6/schemaspy/index.html

Get the attention of the core team and ask someone to update it for the new release.

Consider updating the thread on the mailing list once the update is in place.

See also *SchemaSpy*.

8.18.31 Alert Translators About the New Release

Create an issue at https://github.com/GlobalDataverseCommunityConsortium/dataverse-language-packs/issues to say a new release is out and that we would love for the properties files for English to be added.

For example, for 6.4 we wrote "Update en_US/Bundle.properties etc. for Dataverse 6.4" at https://github.com/GlobalDataverseCommunityConsortium/dataverse-language-packs/issues/125

8.18.32 Add the Release to the Dataverse Roadmap

Add an entry to the list of releases at https://www.iq.harvard.edu/roadmap-dataverse-project

8.18.33 Announce the Release on the Dataverse Blog

Make a blog post at https://dataverse.org/blog

8.18.34 Announce the Release on the Mailing List

Post a message at https://groups.google.com/g/dataverse-community

8.18.35 Announce the Release on Zulip

Post a message under #community at https://dataverse.zulipchat.com

8.18.36 For Hotfixes, Merge Hotfix Branch into "develop"

Note: this only applies to hotfixes!

We've merged the hotfix into the "master" branch but now we need the fixes (and version bump) in the "develop" branch.

Make a new branch off the hotfix branch. You can call it something like "6.7.1-merge-hotfix-to-develop".

In that branch, do the *Update the Container Base Image Version Property* step you skipped above. Now is the time.

Create a pull request against develop. Merge conflicts are possible and this pull request should go through review and QA like normal. Afterwards it's fine to delete this branch and the hotfix branch that was merged into master.

8.18.37 For Hotfixes, Tell Developers to Merge "develop" into Their Branches and Rename SQL Scripts

Note: this only applies to hotfixes!

Because we have merged a version bump from the hotfix into the "develop" branch, any SQL scripts in the "develop" branch should be renamed (from "5.11.0" to "5.11.1" for example). (To read more about our naming conventions for SQL scripts, see *SQL Upgrade Scripts*.)

Look at `src/main/resources/db/migration` in the "develop" branch and if any SQL scripts have the wrong version, make a pull request (or ask a developer to) to update them (all at once in a single PR is fine).

Tell developers to merge the "develop" into their open pull requests (to pick up the new version and any fixes) and rename SQL scripts (if any) with the new version.

8.18.38 Lift the Code Freeze and Encourage Developers to Update Their Branches

It's now safe to lift the code freeze. We can start merging pull requests into the "develop" branch for the next release.

Let developers know that they should merge the latest from the "develop" branch into any branches they are working on. (For hotfixes we've already told them this.)

8.19 Making Library Releases

Contents:

- *Introduction*
- *Maven Central (Java)*
  - *Releasing Existing Libraries to Maven Central*
    - *Releasing a Snapshot Version to Maven Central*
    - *Releasing a Release (Non-Snapshot) Version to Maven Central*
  - *Releasing a New Library to Maven Central*
- *npm (JavaScript/TypeScript)*

8.19.1 Introduction

Note: See *Making Releases* for Dataverse itself.

We release Java libraries to Maven Central that are used by Dataverse (and perhaps other software!):

- https://central.sonatype.com/namespace/org.dataverse
- https://central.sonatype.com/namespace/org.dataverse.test
- https://central.sonatype.com/namespace/io.gdcc
- https://central.sonatype.com/namespace/io.gdcc.export

We release JavaScript/TypeScript libraries to npm:

- https://www.npmjs.com/package/@iqss/dataverse-design-system

8.19.2 Maven Central (Java)

From the perspective of the Maven Central, we are both producers because we publish/release libraries there and consumers because we pull down those libraries (and many others) when we build Dataverse.

Releasing Existing Libraries to Maven Central

If you need to release an existing library, all the setup should be done already. The steps below assume that GitHub Actions are in place to do the heavy lifting for you, such as signing artifacts with GPG.

Releasing a Snapshot Version to Maven Central

Snapshot releases are published automatically through GitHub Actions (e.g. through a snapshot workflow for the SWORD library) every time a pull request is merged (or the default branch, typically `main`, is otherwise updated).

That is to say, to make a snapshot release, you only need to get one or more commits into the default branch.

It's possible, of course, to make snapshot releases outside of GitHub Actions, from environments such as your laptop. Generally, you'll want to look at the GitHub Action and try to do the equivalent. You'll need a file set up locally at `~/.m2/settings.xml` with the following (contact a core developer for the redacted bits):

```
<settings>
  <servers>
    <server>
      <id>central</id>
      <username>REDACTED</username>
      <password>REDACTED</password>
    </server>
  </servers>
</settings>
```

Then, study the GitHub Action and perform similar commands from your local environment. For example, as of this writing, for the dataverse-spi project, you can run the following commands, substituting the suffix you need:

```
mvn -f modules/dataverse-spi -Dproject.version.suffix="2.1.0-PR11767-SNAPSHOT" verify
```

```
mvn -f modules/dataverse-spi -Dproject.version.suffix="2.1.0-PR11767-SNAPSHOT" deploy
```

This will upload the snapshot here, for example: https://central.sonatype.com/repository/maven-snapshots/io/gdcc/dataverse-spi/2.1.02.1.0-PR11767-SNAPSHOT/dataverse-spi-2.1.02.1.0-PR11767-20250827.182026-1.jar

Before OSSRH was retired, you could browse through snapshot jars you published at https://s01.oss.sonatype.org/content/repositories/snapshots/io/gdcc/dataverse-spi/2.0.0-PR9685-SNAPSHOT/, for example. Now, even though you may see the URL of the jar as shown above during the "deploy" step, if you try to browse the various snapshot jars at https://central.sonatype.com/repository/maven-snapshots/io/gdcc/dataverse-spi/2.1.02.1.0-PR11767-SNAPSHOT/ you'll see "This maven2 hosted repository is not directly browseable at this URL. Please use the browse or HTML index views to inspect the contents of this repository." Sadly, the "browse" and "HTML index" links don't work, as noted in a question on the Sonatype Community forum. Below is a suggestion for confirming that the jar was uploaded properly, which is to use Maven to copy the jar to your local directory. You could then compare checksums.

```
mvn dependency:copy -DrepoUrl=https://central.sonatype.com/repository/maven-snapshots/
-Dartifact=io.gdcc:dataverse-spi:2.1.02.1.0-PR11767-SNAPSHOT -DoutputDirectory=.
```

Releasing a Release (Non-Snapshot) Version to Maven Central

From a pom.xml it may not be apparent that snapshots like `6.0-SNAPSHOT` might be changing under your feet. Browsing the snapshot repository (e.g. our UNF 6.0-SNAPSHOT), may reveal versions changing over time. To finalize the code and stop it from changing, we publish/release what Maven calls a "release version". This will remove `-SNAPSHOT` from the version (through an `mvn` command).

Non-snapshot releases (release versions) are published automatically through GitHub Actions (e.g. through a release workflow), kicked off locally by an `mvn` command that invokes the Maven Release Plugin.

First, run a clean:

```
mvn release:clean
```

Then run a prepare:

```
mvn release:prepare
```

The prepare step is interactive. You will be prompted for the following information:

- the release version (e.g. 2.0.0)
- the git tag to create and push (e.g. sword2-server-2.0.0)
- the next development (snapshot) version (e.g. 2.0.1-SNAPSHOT)

These examples from the SWORD library. Below is what to expect from the interactive session. In many cases, you can just hit enter to accept the defaults.

```
[INFO] 5/17 prepare:map-release-versions
What is the release version for "SWORD v2 Common Server Library (forked)"? (sword2-
↪server) 2.0.0: :
[INFO] 6/17 prepare:input-variables
What is the SCM release tag or label for "SWORD v2 Common Server Library (forked)"?␣
↪(sword2-server) sword2-server-2.0.0: :
[INFO] 7/17 prepare:map-development-versions
What is the new development version for "SWORD v2 Common Server Library (forked)"?␣
↪(sword2-server) 2.0.1-SNAPSHOT: :
[INFO] 8/17 prepare:rewrite-poms-for-release
```

Note that a commit or two will be made and pushed but if you do a `git status` you will see that locally you are behind by that number of commits. To fix this, you can just do a `git pull`.

It can take some time for the jar to be visible on Maven Central. You can start by looking on the repo1 server, like this: https://repo1.maven.org/maven2/io/gdcc/sword2-server/2.0.0/

Don't bother putting the new version in a pom.xml until you see it on repo1.

Note that the next snapshot release should be available as well, like this: https://s01.oss.sonatype.org/content/groups/staging/io/gdcc/sword2-server/2.0.1-SNAPSHOT/

Releasing a New Library to Maven Central

At a high level:

- Start with a snapshot release.
- Use an existing pom.xml as a starting point, such as from Croissant, that inherits from the common Maven parent (https://github.com/gdcc/maven-parent). You can also play around with the "hello" project (https://github.com/gdcc/hello) and even make releases from it since it is designed to be a sandbox for publishing to Maven Central.
- Use existing GitHub Actions workflows as a starting point, such as from Croissant. As of this writing we have separate actions for `maven-snapshot.yml` and `maven-release.yml`.
- For repos under https://github.com/IQSS, create secrets in the new library's GitHub repo used by the workflow. This is necessary for the IQSS org because "organization secrets are not available for organizations on legacy per-repository billing plans." For repos under https://github.com/gdcc you can make use of shared secrets at the org level. These are the environment variables we use:
  - DATAVERSEBOT_GPG_KEY
  - DATAVERSEBOT_GPG_PASSWORD
  - DATAVERSEBOT_SONATYPE_TOKEN
  - DATAVERSEBOT_SONATYPE_USERNAME
- If you need an entire new namespace, look at previous issues such as https://issues.sonatype.org/browse/OSSRH-94575 and https://issues.sonatype.org/browse/OSSRH-94577

8.19.3 npm (JavaScript/TypeScript)

Currently, publishing @iqss/dataverse-design-system to npm done manually. We plan to automate this as part of https://github.com/IQSS/dataverse-frontend/issues/140

https://www.npmjs.com/package/js-dataverse is the previous 1.0 version of js-dataverse. No 1.x releases are planned. We plan to publish 2.0 (used by the new frontend) as discussed in https://github.com/IQSS/dataverse-frontend/issues/13

8.20 Metadata Export Formats

Contents:

- *Introduction*
- *Exporter Basics*
- *Building an Exporter*
- *Specifying a Prerequisite Export*

8.20.1 Introduction

Dataverse ships with a number of metadata export formats available for published datasets. A given metadata export format may be available for user download (via the UI and API) and/or be available for use in Harvesting between Dataverse instances.

As of v5.14, Dataverse provides a mechanism for third-party developers to create new metadata Exporters than implement new metadata formats or that replace existing formats. All the necessary dependencies are packaged in an interface JAR file available from Maven Central. Developers can distribute their new Exporters as JAR files which can be dynamically loaded into Dataverse instances - see *External Metadata Exporters*. Developers are encouraged to work with the core Dataverse team (see *Getting Help*) to distribute these JAR files via Maven Central. See the Croissant and Debug artifacts as examples. You may find other examples under *Inventory of External Exporters* in the Installation Guide.

8.20.2 Exporter Basics

New Exports must implement the `io.gdcc.spi.export.Exporter` interface. The interface includes a few methods for the Exporter to provide Dataverse with the format it produces, a display name, format mimetype, and whether the format is for download and/or harvesting use, etc. It also includes a main `exportDataset(ExportDataProvider dataProvider, OutputStream outputStream)` method through which the Exporter receives metadata about the given dataset (via the `ExportDataProvider`, described further below) and writes its output (as an OutputStream).

Exporters that create an XML format must implement the `io.gdcc.spi.export.XMLExporter` interface (which extends the Exporter interface). XMLExporter adds a few methods through which the XMLExporter provides information to Dataverse about the XML namespace and version being used.

Exporters also need to use the `@AutoService(Exporter.class)` which makes the class discoverable as an Exporter implementation.

The `ExportDataProvider` interface provides several methods through which your Exporter can receive dataset and file metadata in various formats. Your exporter would parse the information in one or more of these inputs to retrieve the values needed to generate the Exporter's output format.

The most important methods/input formats are:

- `getDatasetJson()` - metadata in the internal Dataverse JSON format used in the native API and available via the built-in JSON metadata export.
- `getDatasetORE()` - metadata in the OAI_ORE format available as a built-in metadata format and as used in Dataverse's BagIT-based Archiving capability.
- `getDatasetFileDetails` - detailed file-level metadata for ingested tabular files.

The first two of these provide ~complete metadata about the dataset along with the metadata common to all files. This includes all metadata entries from all metadata blocks, PIDs, tags, Licenses and custom terms, etc. Almost all built-in exporters today use the JSON input. The newer OAI_ORE export, which is JSON-LD-based, provides a flatter structure and references metadata terms by their external vocabulary ids (e.g. http://purl.org/dc/terms/title) which may make it a prefereable starting point in some cases.

The last method above provides a new JSON-formatted serialization of the variable-level file metadata Dataverse generates during ingest of tabular files. This information has only been included in the built-in DDI export, as the content of a `dataDscr` element. (Hence inspecting the edu.harvard.iq.dataverse.export.DDIExporter and related classes would be a good way to explore how the JSON is structured.)

The interface also provides

- `getDatasetSchemaDotOrg();` and
- `getDataCiteXml();`.

These provide subsets of metadata in the indicated formats. They may be useful starting points if your exporter will, for example, only add one or two additional fields to the given format.

If an Exporter cannot create a requested metadata format for some reason, it should throw an `io.gdcc.spi.export.ExportException`.

8.20.3 Building an Exporter

The examples at https://github.com/gdcc/exporter-croissant and https://github.com/gdcc/exporter-debug provide a Maven pom.xml file suitable for building an Exporter JAR file and those repositories provide additional development guidance.

There are four dependencies needed to build an Exporter:

- `io.gdcc dataverse-spi` library containing the interfaces discussed above and the ExportException class
- `com.google.auto.service auto-service`, which provides the @AutoService annotation
- `jakarta.json jakarata.json-api` for JSON classes
- `jakarta.ws.rs jakarta.ws.rs-api`, which provides a MediaType enumeration for specifying mime types.

8.20.4 Specifying a Prerequisite Export

An advanced feature of the Exporter mechanism allows a new Exporter to specify that it requires, as input, the output of another Exporter. An example of this is the builting HTMLExporter which requires the output of the DDI XML Exporter to produce an HTML document with the same DDI content.

This is configured by providing the metadata format name via the `Exporter.getPrerequisiteFormatName()` method. When this method returns a non-empty format name, Dataverse will provide the requested format to the Exporter via the `ExportDataProvider.getPrerequisiteInputStream()` method.

Developers and administrators deploying Exporters using this mechanism should be aware that, since metadata formats can be changed by other Exporters, the InputStream received may not hold the expected metadata. Developers should clearly document their compatability with the built-in or third-party Exporters they support as prerequisites.

8.21 Tools

These are handy tools for your *Development Environment*.

Contents:

- *Tools for Faster Deployment*
- *Netbeans Connector Chrome Extension*
- *pgAdmin*
- *Maven*
- *PlantUML*
- *Eclipse Memory Analyzer Tool (MAT)*
- *PageKite*

- *MSV*
- *FontCustom*
- *SonarQube*
- *Infer*
- *lsof*
- *JProfiler*
- *jmap and jstat*

8.21.1 Tools for Faster Deployment

See *IDE Triggered Code Re-Deployments* in the Container Guide.

8.21.2 Netbeans Connector Chrome Extension

The Netbeans Connector extension for Chrome allows you to see changes you've made to HTML pages the moment you save the file without having to refresh your browser. See also http://wiki.netbeans.org/ChromeExtensionInstallation

Unfortunately, while the Netbeans Connector Chrome Extension used to "just work", these days a workaround described at https://www.youtube.com/watch?v=J6lOQS2rWK0&t=130 seems to be necessary. For now, under "Run" (under project properties), choose "Chrome" as the browser rather than "Chrome with NetBeans Connector". After you run the project, click the Netbeans logo in Chrome and then "Debug in NetBeans". For more information, please see the "workaround for Netbeans Connector Chrome Extension" post at https://groups.google.com/d/msg/dataverse-dev/agJZilD1l0Q/cMBkt5KDBQAJ

8.21.3 pgAdmin

You may have installed pgAdmin when following the steps in the *Classic Dev Environment* section but if not, you can download it from https://www.pgadmin.org

8.21.4 Maven

With Maven installed you can run `mvn package` and `mvn test` from the command line. It can be downloaded from https://maven.apache.org

8.21.5 PlantUML

PlantUML is used to create diagrams in the guides and other places. Download it from https://plantuml.com and check out an example script at https://github.com/IQSS/dataverse/blob/v4.6.1/doc/Architecture/components.sh . Note that for this script to work, you'll need the `dot` program, which can be installed on Mac with `brew install graphviz`.

8.21.6 Eclipse Memory Analyzer Tool (MAT)

The Memory Analyzer Tool (MAT) from Eclipse can help you analyze heap dumps, showing you "leak suspects" such as seen at https://github.com/payara/Payara/issues/350#issuecomment-115262625

It can be downloaded from https://www.eclipse.org/mat

If the heap dump provided to you was created with `gcore` (such as with `gcore -o /tmp/app.core $app_pid`) rather than `jmap`, you will need to convert the file before you can open it in MAT. Using `app.core.13849` as example of the original 33 GB file, here is how you could convert it into a 26 GB `app.core.13849.hprof` file. Please note that this operation took almost 90 minutes:

```
/usr/java7/bin/jmap -dump:format=b,file=app.core.13849.hprof /usr/java7/bin/java app.
core.13849
```

A file of this size may not "just work" in MAT. When you attempt to open it you may see something like "An internal error occurred during: "Parsing heap dump from '/tmp/heapdumps/app.core.13849.hprof'". Java heap space". If so, you will need to increase the memory allocated to MAT. On Mac OS X, this can be done by editing `MemoryAnalyzer.app/Contents/MacOS/MemoryAnalyzer.ini` and increasing the value "-Xmx1024m" until it's high enough to open the file. See also https://wiki.eclipse.org/index.php/MemoryAnalyzer/FAQ#Out_of_Memory_Error_while_Running_the_Memory_Analyzer

8.21.7 PageKite

PageKite is a fantastic service that can be used to share your local development environment over the Internet on a public IP address.

With PageKite running on your laptop, the world can access a URL such as http://pdurbin.pagekite.me to see what you see at http://localhost:8080

Sign up at https://pagekite.net and follow the installation instructions or simply download https://pagekite.net/pk/pagekite.py

The first time you run `./pagekite.py` a file at `~/.pagekite.rc` will be created. You can edit this file to configure PageKite to serve up port 8080 (the default app server HTTP port) or the port of your choosing.

According to https://pagekite.net/support/free-for-foss/ PageKite (very generously!) offers free accounts to developers writing software the meets https://opensource.org/docs/definition.php such as the Dataverse Project.

8.21.8 MSV

MSV (Multi Schema Validator) can be used from the command line to validate an XML document against a schema. Download the latest version from https://java.net/downloads/msv/releases/ (msv.20090415.zip as of this writing), extract it, and run it like this:

```
$ java -jar /tmp/msv-20090415/msv.jar Version2-0.xsd ddi.xml
start parsing a grammar.
validating ddi.xml
the document is valid.
```

8.21.9 FontCustom

The custom file type icons were created with the help of *FontCustom <https://github.com/FontCustom/fontcustom>*. Their README provides installation instructions as well as directions for producing your own vector-based icon font.

Here is a vector-based SVG file to start with as a template: `icon-template.svg`

8.21.10 SonarQube

SonarQube is a static analysis tool that can be used to identify possible problems in the codebase, or with new code. It may report false positives or false negatives, but can help identify potential problems before they are reported in prodution or to identify potential causes of problems reported in production.

Download SonarQube from https://www.sonarqube.org and start look in the *bin* directory for a *sonar.sh* script for your architecture. Once the tool is running on http://localhost:9000 you can use it as the URL in this example script to run sonar:

```
#!/bin/sh

mvn sonar:sonar \
-Dsonar.host.url=${your_sonar_url} \
-Dsonar.login=${your_sonar_token_for_project} \
-Dsonar.test.exclusions='src/test/**,src/main/webapp/resources/**' \
-Dsonar.issuesReport.html.enable=true \
-Dsonar.issuesReport.html.location='sonar-issues-report.html' \
-Dsonar.jacoco.reportPath=target/coverage-reports/jacoco-unit.exec
```

Once the analysis is complete, you should be able to access http://localhost:9000/dashboard?id=edu.harvard.iq%3Adataverse to see the report. To learn about resource leaks, for example, click on "Bugs", the "Tag", then "leak" or "Rule", then "Resources should be closed".

8.21.11 Infer

Infer is another static analysis tool that can be downloaded from https://github.com/facebook/infer

Example command to run infer:

```
$  infer -- mvn package
```

Look for "RESOURCE_LEAK", for example.

8.21.12 lsof

If file descriptors are not closed, eventually the open but unused resources can cause problems with system (app servers in particular) stability. Static analysis and heap dumps are not always sufficient to identify the sources of these problems. For a quick sanity check, it can be helpful to check that the number of file descriptors does not increase after a request has finished processing.

For example…

```
$  lsof | grep M6EI0N | wc -l
0
$  curl -X GET "http://localhost:8083/dataset.xhtml?persistentId=doi:10.5072/FK2/M6EI0N"␣
```

(continues on next page)

(continued from previous page)

```
↪> /dev/null
$  lsof | grep M6EI0N | wc -l
500
```

would be consistent with a file descriptor leak on the dataset page.

8.21.13 JProfiler

Tracking down resource drainage, bottlenecks etc gets easier using a profiler.

We thank EJ Technologies for granting us a free open source project license for their Java profiler JProfiler.

8.21.14 jmap and jstat

`jmap` and `jstat` are parts of the standard JDK distribution. jmap allows you to look at the contents of the java heap. It can be used to create a heap dump, that you can then feed to another tool, such as `Memory Analyzer Tool` (see above). It can also be used as a useful tool of its own, for example, to list all the classes currently instantiated in memory:

```
$ jmap -histo <app process id>
```

will output a list of all classes, sorted by the number of instances of each individual class, with the size in bytes. This can be very useful when looking for memory leaks in the application. Another useful tool is `jstat`, that can be used in combination with `jmap` to monitor the effectiveness of garbage collection in reclaiming allocated memory.

In the example script below we stress running Dataverse Software applicatione with GET requests to a specific page in a Dataverse installation, use `jmap` to see how many Dataverse collection, Dataset and DataFile class object get allocated, then run `jstat` to see how the numbers are affected by both "Young Generation" and "Full" garbage collection runs (`YGC` and `FGC` respectively):

(This is script is provided **as an example only**! You will have to experiment and expand it to suit any specific needs and any specific problem you may be trying to diagnose, and this is just to give an idea of how to go about it)

```
#!/bin/sh

# the script takes the numeric id of the app server process as the command line argument:
id=$1

while :
do
    # Access the Dataverse collection xxx 10 times in a row:
    for ((i = 0; i < 10; i++))
    do
       # hide the output, standard and stderr:
       curl http://localhost:8080/dataverse/xxx 2>/dev/null > /dev/null
    done

    sleep 1

    # run jmap and save the output in a temp file:

    jmap -histo $id > /tmp/jmap.histo.out
```

(continues on next page)

(continued from previous page)

```
    # grep the output for Dataverse Collection, Dataset and DataFile classes:
    grep '\.Dataverse$' /tmp/jmap.histo.out
    grep '\.Dataset$' /tmp/jmap.histo.out
    grep '\.DataFile$' /tmp/jmap.histo.out
    # (or grep for whatever else you may be interested in)

    # print the last line of the jmap output (the totals):
    tail -1 /tmp/jmap.histo.out

    # run jstat to check on GC:
    jstat -gcutil ${id} 1000 1 2>/dev/null

    # add a time stamp and a new line:

    date
    echo

done
```

The script above will run until you stop it, and will output something like:

```
439:           141          28200  edu.harvard.iq.dataverse.Dataverse
472:           160          24320  edu.harvard.iq.dataverse.Dataset
674:            60           9600  edu.harvard.iq.dataverse.DataFile
S0     S1     E      O      P     YGC     YGCT    FGC    FGCT     GCT
0.00 100.00  35.32  20.15      ?      7    2.145     0    0.000    2.145
Total     108808814     5909776392
Wed Aug 14 23:13:42 EDT 2019

385:           181          36200  edu.harvard.iq.dataverse.Dataverse
338:           320          48640  edu.harvard.iq.dataverse.Dataset
524:           120          19200  edu.harvard.iq.dataverse.DataFile
S0     S1     E      O      P     YGC     YGCT    FGC    FGCT     GCT
0.00 100.00  31.69  45.11      ?      9    3.693     0    0.000    3.693
Total     167998691     9080163904
Wed Aug 14 23:14:59 EDT 2019

367:           201          40200  edu.harvard.iq.dataverse.Dataverse
272:           480          72960  edu.harvard.iq.dataverse.Dataset
442:           180          28800  edu.harvard.iq.dataverse.DataFile
S0     S1     E      O      P     YGC     YGCT    FGC    FGCT     GCT
0.00 100.00  28.05  69.94      ?     11    5.001     0    0.000    5.001
Total     226826706    12230221352
Wed Aug 14 23:16:16 EDT 2019

... etc.
```

How to analyze the output, what to look for:

First, look at the numbers in the jmap output. In the example above, you can immediately see, after the first three iterations, that every 10 Dataverse installation page loads results in the increase of the number of Dataset classes by 160. I.e., each page load leaves 16 of these on the heap. We can also see that each of the 10 page load cycles increased the heap by roughly 3GB; that each cycle resulted in a couple of YG (young generation) garbage collections, and in

the old generation allocation being almost 70% full. These numbers in the example are clearly quite high and are an indication of some problematic memory allocation by the Dataverse installation page - if this is the result of something you have added to the page, you probably would want to investigate and fix it. However, overly generous memory use **is not the same as a leak** necessarily. What you want to see now is how much of this allocation can be reclaimed by "Full GC". If all of it gets freed by `FGC`, it is not the end of the world (even though you do not want your system to spend too much time running `FGC`; it costs CPU cycles, and actually freezes the application while it's in progress!). It is however a **really** serious problem, if you determine that a growing portion of the old. gen. memory (`"O"` in the `jmap` output) is not getting freed, even by `FGC`. This *is* a real leak now, i.e. something is leaving behind some objects that are still referenced and thus off limits to garbage collector. So look for the lines where the `FGC` counter is incremented. For example, the first `FGC` in the example output above:

```
271:           487          97400  edu.harvard.iq.dataverse.Dataverse
216:          3920          150784  edu.harvard.iq.dataverse.Dataset
337:           372          59520  edu.harvard.iq.dataverse.DataFile
Total     277937182    15052367360
S0     S1     E      O      P     YGC     YGCT    FGC    FGCT     GCT
0.00 100.00  77.66  88.15      ?     17    8.734     0    0.000    8.734
Wed Aug 14 23:20:05 EDT 2019

265:           551         110200  edu.harvard.iq.dataverse.Dataverse
202:          4080         182400  edu.harvard.iq.dataverse.Dataset
310:           450          72000  edu.harvard.iq.dataverse.DataFile
Total     142023031     8274454456
S0     S1     E      O      P     YGC     YGCT    FGC    FGCT     GCT
0.00 100.00  71.95  20.12      ?     22   25.034     1    4.455   29.489
Wed Aug 14 23:21:40 EDT 2019
```

We can see that the first `FGC` resulted in reducing the `"O"` by almost 7GB, from 15GB down to 8GB (from 88% to 20% full). The number of Dataset classes has not budged at all - it has grown by the same 160 objects as before (very suspicious!). To complicate matters, `FGC` does not **guarantee** to free everything that can be freed - it will balance how much the system needs memory vs. how much it is willing to spend in terms of CPU cycles performing GC (remember, the application freezes while `FGC` is running!). So you should not assume that the "20% full" number above means that you have 20% of your stack already wasted and unrecoverable. Instead, look for the next **minium** value of `"O"`; then for the next, etc. Now compare these consecutive miniums. With the above test (this is an output of a real experiment, a particularly memory-hungry feature added to the Dataverse installation page), the minimums sequence (of old. gen. usage, in %) was looking as follows:

```
2.19
2.53
3.00
3.13
3.95
4.03
4.21
4.40
4.64
5.06
5.17
etc. ...
```

It is clearly growing - so now we can conclude that indeed something there is using memory in a way that's not recoverable, and this is a clear problem.

8.22 Universal Numerical Fingerprint (UNF)

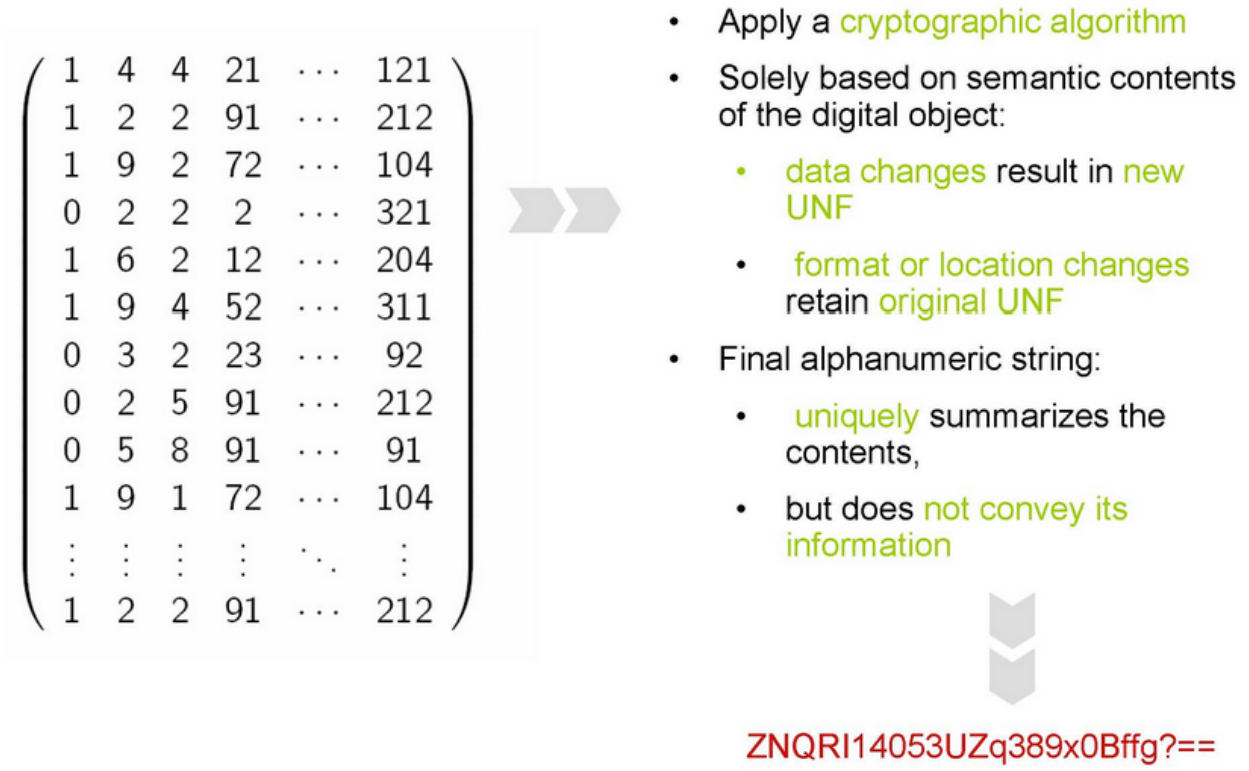


Fig. 1: Fig.1 UNF: used to uniquely identify and verify data.

Universal Numerical Fingerprint (UNF) is a unique signature of the **semantic content** of a digital object. It is **not** simply a checksum of a binary data file. Instead, the UNF algorithm approximates and normalizes the data stored within. A cryptographic hash of that normalized (or canonicalized) representation is then computed. The signature is thus independent of the storage format. E.g., the same data object stored in, say, SPSS and Stata, will have the same UNF.

Early versions of the Dataverse Software were using the first released implementation of the UNF algorithm (v.3, implemented in R). Starting with Dataverse Software 2.0 and throughout the 3.* lifecycle, UNF v.5 (implemented in Java) was used. Dataverse Software 4.0 uses the latest release, UNF v.6. Two parallel implementation, in R and Java, will be available, for cross-validation.

Learn more: Micah Altman and Gary King. 2007. "A Proposed Standard for the Scholarly Citation of Quantitative Data." D-Lib Magazine, 13. Publisher's Version Copy at https://j.mp/2ovSzoT

Contents:

8.22.1 UNF Version 3

Version 3 of the UNF algorithm was used by the Dataverse Network software prior to version 2.0, and was implemented in R code. This algorithm was used on digital objects containing vectors of numbers, vectors of character strings, data sets comprising such vectors, and studies comprising one or more such data sets.

The UNF V3 algorithm applied to the content of a data set or study is as follows:

1. Round each element in a numeric vector to k significant digits using the IEEE 754 round towards zero rounding mode. The default value of k is seven, the maximum expressible in single-precision floating point calculations. UNF calculation for vectors of character strings is identical, except that you truncate to k characters and the default value of k is 128.
2. Convert each vector element to a character string in exponential notation, omitting noninformational zeros. If an element is missing, represent it as a string of three null characters. If an element is an IEEE 754, nonfinite, floating-point special value, represent it as the signed, lowercase, IEEE minimal printable equivalent (that is, +inf, -inf, or +nan).

Each character string comprises the following:

- A sign character.
- A single leading digit.
- A decimal point.
- Up to k-1 digits following the decimal, consisting of the remaining k-1 digits of the number, omitting trailing zeros.
- A lowercase letter "e."
- A sign character.
- The digits of the exponent, omitting trailing zeros.

For example, the number pi at five digits is represented as -3.1415e+, and the number 300 is represented as the string +3.e+2.

1. Terminate character strings representing nonmissing values with a POSIX end-of-line character.
2. Encode each character string with Unicode bit encoding. Versions 3 through 4 use UTF-32BE; Version 4.1 uses UTF-8.
3. Combine the vector of character strings into a single sequence, with each character string separated by a POSIX end-of-line character and a null byte.
4. Compute a hash on the resulting sequence using the standard MD5 hashing algorithm for Version 3 and using SHA256 for Version 4. The resulting hash is base64 encoded to support readability.
5. Calculate the UNF for each lower-level data object, using a consistent UNF version and level of precision across the individual UNFs being combined.
6. Sort the base64 representation of UNFs in POSIX locale sort order.
7. Apply the UNF algorithm to the resulting vector of character strings using k at least as large as the length of the underlying character string.
8. Combine UNFs from multiple variables to form a single UNF for an entire data frame, and then combine UNFs for a set of data frames to form a single UNF that represents an entire research study.

Learn more: Software for computing UNFs is available in an R Module, which includes a Windows standalone tool and code for Stata and SAS languages. Also see the following for more details: Micah Altman and Gary King. 2007. "A Proposed Standard for the Scholarly Citation of Quantitative Data," D-Lib Magazine, Vol. 13, No. 3/4 (March). (Abstract: HTML | Article: PDF)

8.22.2 UNF Version 5

Important Update:

UNF Version 5 has been in use by the Dataverse Project since 2009. It was built into every version of The Dataverse Software, starting with 2.0 and up to 3.6.2. However, some problems were recently found in that implementation. Namely, in certain cases data normalization is not implemented fully to the spec. UNF signatures it generates are still reasonably strong statistically; however, this means that at least some of our signatures are not independently verifiable. I.e., if somebody fully implements their own version of UNF calculator, for certain datasets it would calculate signatures different from those generated by the DVN. Unless of course they implement it with the exact same bugs as ours.

To address this, the Project is about to release UNF Version 6. The release date is still being discussed. It may coincide with the release of Dataverse Software 4.0. Alternatively, the production release of DVN 3.6.3 may get upgraded to use UNF v6 prior to that. This will be announced shortly. In the process, we are solving another problem with UNF v5 - this time we've made an effort to offer very implementer-friendly documentation that describes the algorithm fully and unambiguously. So if you are interested in implementing your own version of a UNF calculator, (something we would like to encourage!) please proceed directly to the Version 6 documentation.

Going forward, we are going to offer a preserved version of the Version 5 library and, possibly, an online UNF v5 calculator, for the purposes of validating vectors and data sets for which published Version 5 UNFs exist.

8.22.3 UNF Version 6

(this document is a draft!)

The document is primarily intended for those who are interested in implementing their own UNF Version 6 calculator. We would like to encourage multiple parallel implementations, since that would be a great (the only, really) way to cross-validate UNF signatures calculated for specific sets of data.

Algorithm Description

UNF v5, on which v6 is based, was originally described in Dr. Micah Altman's paper "A Fingerprint Method for Verification of Scientific Data", Springer Verlag, 2008. The reader is encouraged to consult it for the explanation of the theory behind UNF. However, various changes and clarifications concerning the specifics of normalization have been made to the algorithm since the publication. These crucial details were only documented in the author's unpublished edits of the article and in private correspondence. With this document, a serious effort has been made to produce a complete step-by-step description of the entire process. It should be fully sufficient for the purposes of implementing the algorithm.

Contents:

- *I. UNF of a Data Vector*
- *II. Combining multiple UNFs to create UNFs of higher-level objects.*
- *Footnotes:*

I. UNF of a Data Vector

For each individual vector in a data frame, calculate its UNF signature as follows:

Ia. Normalize each vector element as follows:

1. For a vector of numeric elements: Round each vector element to `N` significant digits using the `IEEE 754` "round towards nearest, ties to even" rounding mode. The default value of `N` is 7.

(See an Important *Note* on the use of *default and optional* values and methods!)

Convert each vector element into a character string in exponential notation, as follows:

A sign character.
A single leading non-zero digit.
A decimal point.
Up to `N-1` remaining digits following the decimal, no trailing zeros.
A lowercase letter "`e`".
A sign character.
The digits of the exponent, omitting trailing zeros.

Special cases:

Zero representation (an exception to the "leading non-zero digit" rule, above):
`+0.e+` for positive zero.
`-0.e+` for negative zero.
(see the *Note* below on *negative zero*)

`Infinity` and `NaN` ("Not a Number") values:

If an element is an `IEEE 754`, non-finite, special floating-point value, represent it as the signed, lowercase, IEEE minimal printable equivalent, that is, `+inf`, `-inf`, or `+nan`. No attempt is made to differentiate between various types of NaNs allowed under `IEEE 754`.

Examples:

The number `1` is normalized as `+1.e+`
The number `pi` at 5 digits is normalized as `+3.1415e+`
The number `-300` is normalized as `-3.e+2`
The number `0.00073` is normalized as `+7.3e-4`
Positive infinity is normalized as `+inf`
The number `1.23456789`, normalized with the default rounding to 7 digits of precision, is normalized as `+1.234568e+`

An "official" list of pre-calculated sample UNFs is supplied with the source of the Java implementation of UNF v6; see the *Note* at the end of the document.

2. For a vector of character strings:

Encode each character string with Unicode bit encoding. In UNF Version 6 UTF-8 is used. Truncate each string to `X` characters; the default value of `X` is 128. No further normalization is performed.

3. Vectors of Boolean values

Should be treated as numeric vectors of `0` s and `1` s.

4. Bit fields.

Normalize bit fields by converting to big-endian form, truncating all leading empty bits, aligning to a byte boundary by padding with leading zero bits, and `base64` encoding to form a character string representation.

5. Normalize dates, times and intervals as follows:

5a. Dates.

Convert calendar dates to a character string of the form `YYYY-MM-DD`, zero padded. Partial dates in the form `YYYY` or `YYYY-MM` are permitted

5b. Time.

Time representation is based on an `ISO 8601` format `hh:mm:ss.fffff`. `hh`, `mm` and `ss` are 2 digit, zero-padded numbers. `fffff` represents fractions of a second, it must contain no trailing (non-significant) zeroes, and must be omitted altogether the value is zero. No other fractional representations, such as fractional minutes or hours, are permitted. If the time zone of the observation is known, convert the time value to the UTC time zone and append a "`Z`" to the time representation. (In other words, no time zones other than UTC are allowed in the final normalized representation).

(see the *Note* at the end of this document for a discussion on *potential issues when calculating UNFs of time values*).

5c. Combined Date and Time values.

Format elements that comprise a combined date and time by concatenating the (full) date representation, a single letter "`T`", and the time representation. Partial date representations are **prohibited** in combined date and time values.

5d. Intervals.

Represent intervals by using two date-time values, each formatted as defined previously, and separated by a slash ("`/`").

Durations, that were mentioned in the old UNF v5 document are NOT in fact implemented and have been dropped from the spec.

Examples:

`2:29 pm on Jun 10, 2012` is normalized as "`2012-06-10T14:29:00`".

`Fri Aug 22 12:51:05 EDT 2014` is normalized as "`2014-08-22T16:51:05Z`"
(The UTC offset of Eastern Daylight Time is -4:00).

6. Missing values Missing values, of all of the above types, are encoded as 3 null bytes: `\000\000\000`.

Ib. Calculate the UNF of the vector as follows:

Terminate each character string representing a NON-MISSING value with a `POSIX` end-of-line character and a null byte (\000). Do not terminate missing value representations (3 null bytes \000\000\000). Concatenate all the individual character strings, and compute the `SHA256` hash of the combined string. Truncate the resulting hash to `128` bits (128 being the default, with other values possible - see the note at the end of the document). Encode the resulting string in `base64`, for readability. Prepend the encoded hash string with the signature header `UNF:6:` (with `6` indicating the current version).

Example:

Vector (numeric): `{1.23456789, <MISSING VALUE>, 0}`
Normalized elements (`N = 7`,default): "`+1.234568e+`", "`\000\000\000`", "`+0.e+`"
Combined string: "`+1.234568e+\n\000\000\000\000+0.e+\n\000`"
`SHA256` hash. truncated to the default `128` bits: `Do5dfAoOOFt4FSj0JcByEw==`
Printable UNF: `UNF:6:Do5dfAoOOFt4FSj0JcByEw==`

II. Combining multiple UNFs to create UNFs of higher-level objects.

IIa. Combine the UNFs of multiple variables to form the UNF for an entire data frame as follows:

UNF of a data frame (datafile) with 1 variable:
The UNF of the data frame is the same as the UNF of the variable.

UNF of a data frame with the number of variables > 1:
Sort the printable UTF8 representations of the individual UNFs in the POSIX locale sort order.
Apply the UNF algorithm to the resulting vector of character strings.

Do note the **sorting** part, above, it is important! In a vector of observations, the order is important; changing the order of observations changes the UNF. A data frame, however, is considered an unordered set of individual vectors. I.e., re-arranging the order in which data variable columns occur in an R or Stata file should not affect the UNF. Hence the UNFs of individual variables are sorted, before the combined UNF of the data frame is calculated.

IIb. Similarly, combine the UNFs for a set of data frames to form a single UNF that represents an entire research study ("dataset").

Again, the UNF of a study (dataset) with a single file = the UNF of the file; for more than one file, calculate the study UNF as described above.

Using a consistent UNF version and level of precision across an entire dataset is recommended when calculating the UNFs of individual data objects.

Footnotes:

Note: On default and non-default parameter values: Here and throughout the rest of this document, phrases like "The default value of `N` is 7" suggest that it is possible to use non-default values, such as a different number of digits of precision, in this case. This has been a source of some confusion in the past. UNF relies on data normalization to produce "data fingerprints" that are meaningful and descriptive. So how do you generate reproducible and verifiable signatures if any flexibility is allowed in the normalization algorithm? The answer, as specified in the original UNF paper: any non-default parameters used are embedded in the header portion of the UNF!

For example, to specify a non-default precision the parameter it is specified using the parameter `N`, formatted as follows:

`Nnnn` - where `nnn` is the number of precision digits, different from the default `7`.

Example:
The UNF of a floating point (Double) vector with a single element of `1.23456789`, calculated with the default 7 digits of precision, is `UNF:6:vcKELUSS4s4k1snF4OTB9A==`. If we want to calculate the signature with `N = 9`, the resulting printable UNF is `UNF:6:N9:IKw+l4ywdwsJeDze8dplJA==`. With the parameter value embedded in the signature, it can be recalculated and verified unambiguously.

Other optional parameters supported:
(**multiple parameters are added comma-separated, in any order**)

`X###` - where `###` is the number of bytes for truncation of character strings;
`128` is the default.
`H###` - where `###` is the number of bits to which the `SHA256` hash should be truncated.
Allowed values are {`128` , `192` , `196` , `256`} with `128` being the default.
`R1` - **truncate** numeric values to `N` digits, **instead of rounding**, as previously described.

Dr. Micah Altman's classic UNF v5 paper mentions another optional parameter `T###`, for specifying rounding of date and time values (implemented as stripping the values of entire components - fractional seconds, seconds, minutes, hours… etc., progressively) - but it doesn't specify its syntax. It is left as an exercise for a curious reader to contact the author and work out the details, if so desired. (Not implemented in UNF Version 6 by the Dataverse Project).

Note: we do not recommend truncating character strings at fewer bytes than the default `128` (the `X` parameter). At the very least this number **must** be high enough so that the printable UNFs of individual variables or files are not truncated, when calculating combined UNFs of files or datasets, respectively.

It should also be noted that the Dataverse Software never calculates UNFs with any non-default parameters. And we are not aware of anyone else actually doing so. If you are considering creating your own implementation of the UNF, it may be worth trying to create a simplified, defaults-only version first. Such an implementation would be sufficient to independently verify Dataverse Software-produced UNFs, among other things.

Note: Negative Zero

`IEEE 754` zero is signed. I.e., there are 2 zeros, positive and negative. As implemented in most programming languages, floating point types can have negative zero values. It is the responsibility of the implementer, to properly identify the sign of a floating point zero value. Which can be a bit tricky; for example, in Java programming language, when performing arithmetic comparison on values of the primitive type `double`, the following evaluates to `TRUE`:
`0.0d == -0.0d`

However, Java also provides a wrapper class `Double`, with comparison methods that recognize `-0.0` and `0.0` as different values, and `0.0` to be greater than `-0.0`. So all of the following expressions evaluate to `FALSE`:

```
new Double(0.0d).equals(new Double(-0.0d))
Double.compare(-0.0d, 0.0d) >= 0
new Double(-0.0d).compareTo(new Double(0.0d)) >= 0
```

Note: UNFs of time values in real-life statistical packages

The following is not by itself an implementation concern. But it is something you may need to consider when calculating UNFs of time values from real-world data.

The fact that the same time value with and without the time zone specified produces different UNFs presents an interesting issue when converting data between different formats. For example, in STATA none of the available time types support time zones. In R, on the other hand, ALL time values are stored with a time zone. While it is possible to create an R time value from a character representation with no time zone - for example:

```
timevar<-as.POSIXct("03/19/2013 18:20:00", format = "%m/%d/%Y %H:%M:%OS");
```

it still results in R assuming the time is in the **current** time zone, and storing the UTC equivalent of that time. In fact R always stores its time values in UTC; specific time zones can be defined, as attributes, in which case the values will be adjusted accordingly for display. Otherwise the display representation will be readjusted each time the vector is viewed, according to the time zone **current to the viewer**. Meaning that the human readable representation of the same stored time value will be different when viewed on systems in different time zones. With that in mind, it appears that the only way to calculate a meaningful UNF of a time value from an R data frame is to use the stored UTC time - resulting in the "Z" in the normalized string. And that further means that it is impossible to convert a data frame with time values from STATA to R, or the other way around, and have the same UNF preserved.

We do not consider this a problem with the algorithm. These differences between the two approaches to handling time values, in R and STATA, should in fact be considered as **significant**. Enough so to conclude that the format conversion actually changes the data **semantically**. Which, in turn, justifies a new UNF.

If for whatever reason it is important to produce an R version of a STATA file while preserving the UNF, it can still be done. One way to achieve that would be to convert the original time vector to a String vector in R, in the format identical to that used in the UNF normalization algorithm, e.g., "`yy-mm-ddThh:mm:ss`". One would not be able to use this resulting R vector in any time-based calculations without extra type conversion. But the data frame would produce the same UNF.

More UNF Examples:

An "official" list of sample UNFs of various data types is provided with the source of the UNF v6 Java implementation.

8.23 Make Data Count

Support for Make Data Count is a feature of the Dataverse Software that is described in the *Make Data Count* section of the Admin Guide. In order for developers to work on the feature, they must install Counter Processor, a Python 3 application, as described below. Counter Processor can be found at https://github.com/gdcc/counter-processor

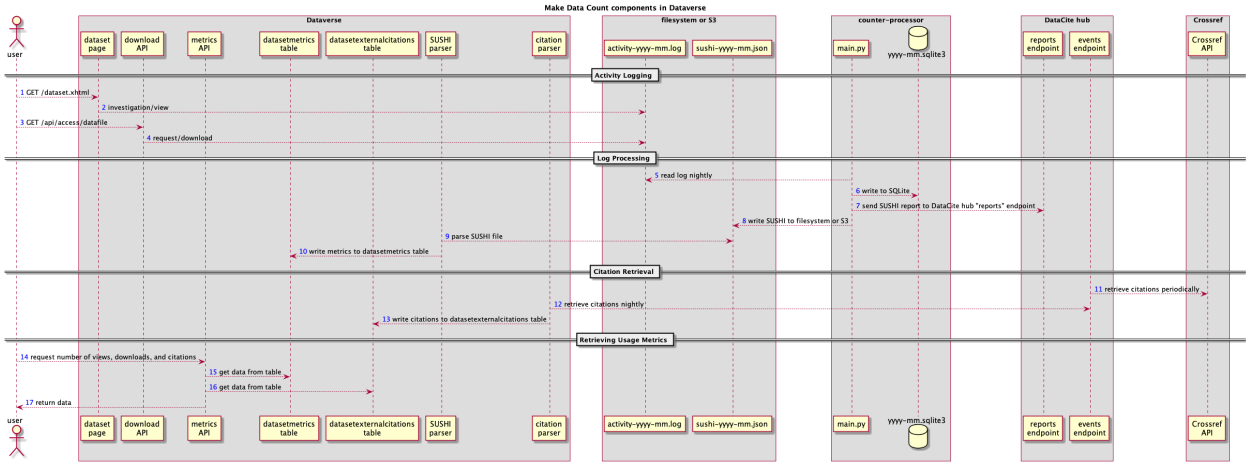
Contents:

- *Architecture*
- *Dev Environment Setup for Make Data Count*
  - *Generate Fake Metrics Only*

- *Full Setup*
- *Testing Make Data Count and Your Dataverse Installation*
- *Processing Archived Logs*
- *Resources*

8.23.1 Architecture

There are many components involved in the Dataverse Software's architecture for Make Data Count as shown in the diagram below.



8.23.2 Dev Environment Setup for Make Data Count

Generate Fake Metrics Only

The quickest way to put populate tables with some data is to run the `MakeDataCountApiIT` integration test. For more on running integration tests see the *Testing* section. This will populate views and downloads.

To insert a citation you could insert a row like below, changing "72" in the example below with the dataset id for your dataset.

```
INSERT INTO datasetexternalcitations (id, citedbyurl, dataset_id) VALUES (1, 'https://
doi.org/10.1007/s12520-017-0578-2', 72);
```

Full Setup

The recommended way to work on the Make Data Count feature is to spin up an EC2 instance that has both the Dataverse Software and Counter Processor installed. Go to the *Deployment* page for details on how to spin up an EC2 instance and make sure that your Ansible file is configured to install Counter Processor before running the "create" script.

After you have spun to your EC2 instance, set `:MDCLogPath` so that the Dataverse installation creates a log for Counter Processor to operate on. For more on this database setting, see the *Configuration* section of the Installation Guide.

Next you need to have the Dataverse installation add some entries to the log that Counter Processor will operate on. To do this, click on some published datasets and download some files.

Next you should run Counter Processor to convert the log into a SUSHI report, which is in JSON format. Before running Counter Processor, you need to put a configuration file into place. As a starting point use `counter-processor-config.yaml` and edit the file, paying particular attention to the following settings:

- `log_name_pattern` You might want something like `/usr/local/payara7/glassfish/domains/domain1/logs/counter_(yyyy-mm-dd).log`
- `year_month` You should probably set this to the current month.
- `output_file` This needs to be a directory that the "dataverse" Unix user can read but that the "counter" user can write to. In dev, you can probably get away with "/tmp" as the directory.
- `platform` Out of the box from Counter Processor this is set to `Dash` but this should be changed to match the name of your Dataverse installation. Examples are "Harvard Dataverse Repository" for Harvard University or "LibraData" for the University of Virginia.
- `upload_to_hub` This should be "False" unless you are testing sending SUSHI reports to the DataCite hub.
- `simulate_date` You should probably set this to tomorrow.

Once you are done with your configuration, you can run Counter Processor like this:

```
sudo -i
```

```
su - counter
```

```
cd /usr/local/counter-processor-1.06
```

```
CONFIG_FILE=counter-processor-config.yaml python39 main.py
```

(Please note that the Counter Processor README says you can also pass in values like `START_DATE`, `END_DATE` etc. at the command line if you find this to be more convenient.)

After `main.py` has finished, you should see output that a JSON file has been placed in the directory you specified in `output_file`. Next, pass this JSON file to curl like this, substituting the DOI of the dataset you're testing with:

```
curl -X POST 'http://localhost:8080/api/admin/makeDataCount/:persistentId/
addUsageMetricsFromSushiReport?reportOnDisk=/tmp/sushi_sample_logs.
json&persistentId=doi:10.5072/FK2/BL2IBM
```

(Note that in production the persistentId of a dataset is not passed. Instead the PIDs for the datasets are pulled out of the JSON file.)

Assuming the curl command above worked, you should be able to retrieve the views and downloads via API like this, substituting the metric you're interested in (viewsTotal, viewsUnique, downloadsTotal, downloadsUnique) and the DOI of the dataset you're testing with:

```
curl -X POST 'http://localhost:8080/api/datasets/:persistentId/makeDataCount/viewsTotal/
2019-01&persistentId=doi:10.5072/FK2/BL2IBM
```

If all this is working and you want to send data to the test instance of the Datacite hub, change `upload_to_hub` to "True" and contact support@datacite.org to get a JSON Web Token (JWT) to test with. Counter Processor should send the SUSHI reports for you but if you need to troubleshoot sending the reports manually, you can try the following curl command, substituting your JWT:

```
curl --header "Content-Type:  application/json; Accept:  application/json" -H
"Authorization:  Bearer $JSON_WEB_TOKEN" -X POST https://api.test.datacite.org/reports/
-d @sushi_report.json
```

For how to put citations into your dev database and how to get them out again, see *Configuring Your Dataverse Installation for Make Data Count Citations* section in Make Data Count of the Admin Guide.

8.23.3 Testing Make Data Count and Your Dataverse Installation

A developer running Counter Processor alongside the Dataverse installation for development or testing purposes will notice that once the raw Dataverse installation logs have been processed, there is no straightforward way to re-test those same logs.

The first thing to fix is to clear two files from Counter Processor `state` folder, `statefile.json` and `counter_db_[yyyy-mm].sqlite3`

Second, if you are also sending your SUSHI report to Make Data Count, you will notice that re-running Counter Processor will not update the information logged on their servers. This is due to us clearing the state of Counter Processor, which in turn causes Counter Processor to send incorrectly structured requests to Make Data Count. The easiest way to resolve this issue is to `DELETE` the record Counter Processor has created on Make Data Count:

```
curl -H "Authorization: Bearer $JSON_WEB_TOKEN" -X DELETE https://$MDC_SERVER/reports/
$REPORT_ID
```

To get the `REPORT_ID`, look at the logs generated in `/usr/local/counter-processor-1.06/tmp/datacite_response_body.txt`

To read more about the Make Data Count api, see https://github.com/datacite/sashimi

You can compare the MDC metrics display with the Dataverse installation's original by toggling the `:DisplayMDCMetrics` setting (true by default to display MDC metrics).

8.23.4 Processing Archived Logs

A new script (release date TBD) will be available for processing archived Dataverse log files. Monthly logs that are zipped, TARed, and copied to an archive can be processed by this script running nightly or weekly.

The script will keep track of the state of each tar file they are processed and will make use of the following "processingState" API endpoints, which allow the state of each file to be checked or modified.

The possible states are new, done, skip, processing, and failed.

Setting the state to "skip" will prevent the file from being processed if the developer needs to analyze the contents.

"failed" files will be re-tried in a later run.

"done" files are successful and will be ignored going forward.

The files currently being processed will have the state "processing".

The script will process the newest set of log files (merging files from multiple nodes) and call Counter Processor.

APIs to manage the states include GET, POST, and DELETE (for testing), as shown below.

Note: `yearMonth` must be in the format `yyyymm` or `yyyymmdd`. Note: If running the new script on multiple servers add the query parameter &server=serverName on the first POST call. The server name can not be changed once set. To clear the name out you must delete the state and post a new one.

```
curl -X GET http://localhost:8080/api/admin/makeDataCount/{yearMonth}/processingState
```

```
curl -X POST http://localhost:8080/api/admin/makeDataCount/{yearMonth}/processingState?
state=processing&server=server1              curl -X POST http://localhost:8080/api/admin/
makeDataCount/{yearMonth}/processingState?state=done
```

```
curl -X DELETE http://localhost:8080/api/admin/makeDataCount/{yearMonth}/processingState
```

8.23.5 Resources

The original issue for adding Make Data Count support is https://github.com/IQSS/dataverse/issues/4821

8.24 Shibboleth, OAuth and OIDC

Contents:

- *Shibboleth and OAuth*
- *OpenID Connect (OIDC)*

8.24.1 Shibboleth and OAuth

If you are working on anything related to users, please keep in mind that your changes will likely affect Shibboleth and OAuth users. For some background on user accounts in the Dataverse Software, see *Auth Modes: Local vs. Remote vs. Both* section of Configuration in the Installation Guide.

Rather than setting up Shibboleth on your laptop, developers are advised to add the Shibboleth auth provider (see "Add the Shibboleth Authentication Provider to Your Dataverse Installation" at *Shibboleth*) and add a value to their database to enable Shibboleth "dev mode" like this:

```
curl http://localhost:8080/api/admin/settings/:DebugShibAccountType -X PUT -d RANDOM
```

For a list of possible values, please "find usages" on the settings key above and look at the enum.

Now when you go to http://localhost:8080/shib.xhtml you should be prompted to create a Shibboleth account.

OAuth is much more straightforward to get working on your laptop than Shibboleth. GitHub is a good identity provider to test with because you can easily request a Client ID and Client Secret that works against localhost. Follow the instructions in the *OAuth Login Options* section of the installation Guide and use "http://localhost:8080/oauth2/callback.xhtml" as the callback URL.

In addition to setting up OAuth on your laptop for real per above, you can also use a dev/debug mode:

```
curl http://localhost:8080/api/admin/settings/:DebugOAuthAccountType -X PUT -d
RANDOM_EMAIL2
```

For a list of possible values, please "find usages" on the settings key above and look at the enum.

Now when you go to http://localhost:8080/oauth2/firstLogin.xhtml you should be prompted to create an OAuth account.

8.24.2 OpenID Connect (OIDC)

STOP! `oidc-keycloak-auth-provider.json` was changed from http://localhost:8090 to http://keycloak.mydomain.com:8090 to test *Using Preview URL Tokens as API Token*. In addition, `docker-compose-dev.yml` in the root of the repo was updated to start up Keycloak. To use these, you should add `127.0.0.1 keycloak.mydomain.com` to your `/etc/hosts file`. If you'd like to use the docker compose as described below (`conf/keycloak/docker-compose.yml`), you should revert the change to `oidc-keycloak-auth-provider.json`.

If you are working on the OpenID Connect (OIDC) user authentication flow, you do not need to connect to a remote provider (as explained in *OpenID Connect Login Options*) to test this feature. Instead, you can use the available configuration that allows you to run a test Keycloak OIDC identity management service locally through a Docker container.

(Please note! The client secret (`94XHrfNRwXsjqTqApRrwWmhDLDHpIYV8`) is hard-coded in `test-realm.json` and `oidc-keycloak-auth-provider.json`. Do not use this config in production! This is only for developers.)

You can find this configuration in `conf/keycloak`. There are two options available in this directory to run a Keycloak container: bash script or docker-compose.

To run the container via bash script, execute the following command (positioned in `conf/keycloak`):

```
./run-keycloak.sh
```

The script will create a Keycloak container or restart it if the container was already created and stopped. Once the script is executed, Keycloak should be accessible from http://localhost:8090/

Now load the configuration defined in `oidc-keycloak-auth-provider.json` into your Dataverse installation to enable Keycloak as an authentication provider.

```
curl -X POST -H 'Content-type: application/json' --upload-file oidc-keycloak-auth-provider.
json http://localhost:8080/api/admin/authenticationProviders
```

You should see the new provider, called "OIDC-Keycloak", under "Other options" on the Log In page.

You should be able to log into Keycloak with the one of the following credentials:

Username	Password
admin	admin
curator	curator
user	user
affiliate	affiliate

In case you want to stop and remove the Keycloak container, just run the other available bash script:

```
./rm-keycloak.sh
```

Note: the Keycloak admin to login at the admin console is `kcadmin:kcpassword`

8.25 Geospatial Data

Contents:

- *How The Dataverse Software Ingests Shapefiles*
  - *Ingest*
  - *Example*

8.25.1 How The Dataverse Software Ingests Shapefiles

A shapefile is a set of files, often uploaded/transferred in `.zip` format. This set may contain up to fifteen files. A minimum of three specific files (`.shp`, `.shx`, `.dbf`) are needed to be a valid shapefile and a fourth file (`.prj`) is required for some applications – or any type of meaningful visualization.

For ingest, four files are the minimum required:

- `.shp` - shape format; the feature geometry itself
- `.shx` - shape index format; a positional index of the feature geometry to allow seeking forwards and backwards quickly
- `.dbf` - attribute format; columnar attributes for each shape, in dBase IV format
- `.prj` - projection format; the coordinate system and projection information, a plain text file describing the projection using well-known text format

Ingest

When uploaded to a Dataverse installation, the `.zip` is unpacked (same as all `.zip` files). Shapefile sets are recognized by the same base name and specific extensions. These individual files constitute a shapefile set. The first four are the minimum required (`.shp`, `.shx`, `.dbf`, `.prj`)

For example:

- bicycles.shp (required extension)
- bicycles.shx (required extension)
- bicycles.prj (required extension)
- bicycles.dbf (required extension)
- bicycles.sbx (NOT required extension)
- bicycles.sbn (NOT required extension)

Upon recognition of the four required files, the Dataverse installation will group them as well as any other relevant files into a shapefile set. Files with these extensions will be included in the shapefile set:

- Required: `.shp`, `.shx`, `.dbf`, `.prj`
- Optional: `.sbn`, `.sbx`, `.fbn`, `.fbx`, `.ain`, `.aih`, `.ixs`, `.mxs`, `.atx`, `.cpg`, `.qpj`, `.qmd`, `shp.xml`

Then the Dataverse installation creates a new `.zip` with mimetype as a shapefile. The shapefile set will persist as this new `.zip`.

Example

1a. Original `.zip` contents:

A file named `bikes_and_subways.zip` is uploaded to the Dataverse installation. This `.zip` contains the following files.

- `bicycles.shp` (shapefile set #1)
- `bicycles.shx` (shapefile set #1)
- `bicycles.prj` (shapefile set #1)
- `bicycles.dbf` (shapefile set #1)

- `bicycles.sbx` (shapefile set #1)
- `bicycles.sbn` (shapefile set #1)
- `bicycles.txt`
- `the_bikes.md`
- `readme.txt`
- `subway_line.shp` (shapefile set #2)
- `subway_line.shx` (shapefile set #2)
- `subway_line.prj` (shapefile set #2)
- `subway_line.dbf` (shapefile set #2)

1b. The Dataverse installation unzips and re-zips files:

Upon ingest, the Dataverse installation unpacks the file `bikes_and_subways.zip`. Upon recognizing the shapefile sets, it groups those files together into new `.zip` files:

- files making up the "bicycles" shapefile become a new `.zip`
- files making up the "subway_line" shapefile become a new `.zip`
- remaining files will stay as they are

To ensure that a shapefile set remains intact, individual files such as `bicycles.sbn` are kept in the set – even though they are not used for mapping.

1c. The Dataverse installation final file listing:

- `bicycles.zip` (contains shapefile set #1: `bicycles.shp`, `bicycles.shx`, `bicycles.prj`, `bicycles.dbf`, `bicycles.sbx`, `bicycles.sbn`)
- `bicycles.txt` (separate, not part of a shapefile set)
- `the_bikes.md` (separate, not part of a shapefile set)
- `readme.txt` (separate, not part of a shapefile set)
- `subway_line.zip` (contains shapefile set #2: `subway_line.shp`, `subway_line.shx`, `subway_line.prj`, `subway_line.dbf`)

For two "final" shapefile sets, `bicycles.zip` and `subway_line.zip`, a new mimetype is used:

- Mimetype: `application/zipped-shapefile`
- Mimetype Label: "Shapefile as ZIP Archive"

8.26 SELinux

Contents:

- *Introduction*
- *Development Environment*
- *Recreating the shibboleth.te File*
  - *Ensure that SELinux is Enforcing*

- *Removing the Existing shibboleth.te Rules*
- *Exercising SELinux denials*
- *Stub out the new shibboleth.te file*
- *Iteratively Use audit2allow to Add Rules and Test Your Change*

8.26.1 Introduction

The `shibboleth.te` file below that was mentioned in the *Shibboleth* section of the Installation Guide was created on CentOS 6 as part of https://github.com/IQSS/dataverse/issues/3406 but may need to be revised for future versions of RHEL/CentOS (pull requests welcome!). The file is versioned with the docs and can be found in the following location:

`doc/sphinx-guides/source/_static/installation/files/etc/selinux/targeted/src/policy/domains/misc/shibboleth.te`

```
module shibboleth 1.0;

require {
    class file {open read};
    class sock_file write;
    class unix_stream_socket connectto;
    type httpd_t;
    type initrc_t;
    type var_run_t;
    type var_t;
}

allow httpd_t initrc_t:unix_stream_socket connectto;
allow httpd_t var_run_t:sock_file write;
allow httpd_t var_t:file {open read};
```

This document is something of a survival guide for anyone who is tasked with updating this file.

8.26.2 Development Environment

In order to work on the `shibboleth.te` file you need to `ssh` into a RHEL or CentOS box running Shibboleth (instructions are in the *Shibboleth* section of the Installation Guide) such as https://beta.dataverse.org or https://demo.dataverse.org that has all the commands below installed. As of this writing, the `policycoreutils-python` RPM was required.

8.26.3 Recreating the shibboleth.te File

If you're reading this page because someone has reported that Shibboleth doesn't work with SELinux anymore (due to an operating system upgrade, perhaps) you *could* start with the existing `shibboleth.te` file, but it is recommended that you create a new one instead to ensure that extra lines aren't included that are no longer necessary.

The file you're recreating is called a Type Enforcement (TE) file, and you can read more about it at https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/chap-Security-Enhanced_Linux-SELinux_Contexts.html

The following doc may or may not be helpful to orient you: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/sect-Security-Enhanced_Linux-Fixing_Problems-Allowing_Access_audit2allow.html

Ensure that SELinux is Enforcing

If `getenforce` returns anything other than `Enforcing`, run `setenforce Enforcing` or otherwise configure SELinux by editing `/etc/selinux/config` and rebooting until SELinux is enforcing.

Removing the Existing shibboleth.te Rules

Use `semodule -l | grep shibboleth` to see if the `shibboleth.te` rules are already installed. Run `semodule -r shibboleth` to remove the module, if necessary. Now we're at square one (no custom rules) and ready to generate a new `shibboleth.te` file.

Exercising SELinux denials

As of this writing, the only component of the Dataverse Software which is known not to work with SELinux out of the box is Shibboleth.

We will be exercising SELinux denials with Shibboleth, and the SELinux-related issues are expected out the box:

- Problems with the dropdown of institutions being created on the Login Page ("Internal Error - Failed to download metadata from /Shibboleth.sso/DiscoFeed.").
- Problems with the return trip after you've logged into HarvardKey or whatever ("shibsp::ListenerException" and "Cannot connect to shibd process, a site adminstrator should be notified.").

In short, all you need to do is try to log in with Shibboleth and you'll see problems associated with SELinux being enabled.

Stub out the new shibboleth.te file

Iterate on the new `shibboleth.te` file wherever you like, such as the root user's home directory in the example below. Start by adding a `module` line like this:

```
echo 'module shibboleth 1.0;' > /root/shibboleth.te
```

Note that a version is required and perhaps it should be changed, but we'll stick with `1.0` for now. The point is that the `shibboleth.te` file must begin with that "module" line or else the `checkmodule` command you'll need to run later will fail. Your file should look like this:

```
module shibboleth 1.0;
# require lines go here
# allow lines go here
```

Iteratively Use audit2allow to Add Rules and Test Your Change

Now that `shibboleth.te` has been stubbed out, we will iteratively add lines to it from the output of piping SELinux Access Vector Cache (AVC) denial messages to `audit2allow -r`. These errors are found in `/var/log/audit/audit.log` so tail the file as you attempt to log in to Shibboleth.

```
# tail -f /var/log/audit/audit.log | fgrep type=AVC
```

You should see messages that look something like this:

```
type=AVC msg=audit(1476728970.378:271405):  avc:  denied { write } for pid=28548
comm="httpd" name="shibd.sock" dev=dm-2 ino=393300 scontext=unconfined_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:var_run_t:s0 tclass=sock_file
```

Next, pipe these message to `audit2allow -r` like this:

```
echo 'type=AVC msg=audit(1476728970.378:271405):  avc:  denied { write
} for pid=28548 comm="httpd" name="shibd.sock" dev=dm-2 ino=393300
scontext=unconfined_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:var_run_t:s0
tclass=sock_file' | audit2allow -r
```

This will produce output like this:

```
require {
        type var_run_t;
        type httpd_t;
        class sock_file write;
}

#============= httpd_t ==============
allow httpd_t var_run_t:sock_file write;
```

Copy and paste this output into the `shibboleth.te` file you stubbed out above. Then, use the same `checkmodule`, `semodule_package`, and `semodule` commands documented in the *Shibboleth* section of the Installation Guide on your file to activate the SELinux rules you're constructing.

Once your updated SELinux rules are in place, try logging in with Shibboleth again. You should see a different AVC error. Pipe that error into `audit2allow -r` as well and put the resulting content into the `shibboleth.te` file you're constructing. As you do this, manually reformat the file using the following rules:

- Put the `require` block at the top.
- Within the require block, sort the lines.
- Put the `allow` lines at the bottom and sort them.
- Where possible, avoid duplicate lines by combining operations such as `open` and `read` into `{open read}`.
- Remove all comment lines.

Keep iterating until it works and then create a pull request based on your updated file. Good luck!

Many thanks to Bill Horka from IQSS for his assistance in explaining how to construct a SELinux Type Enforcement (TE) file!

8.27 Auxiliary File Support

Auxiliary file support is experimental and as such, related APIs may be added, changed or removed without standard backward compatibility. Auxiliary files in the Dataverse Software are being added to support depositing and downloading differentially private metadata, as part of the OpenDP project. In future versions, this approach will likely become more broadly used and supported.

8.27.1 Adding an Auxiliary File to a Datafile

To add an auxiliary file, specify the primary key of the datafile (FILE_ID), and the formatTag and formatVersion (if applicable) associated with the auxiliary file. There are multiple form parameters. "Origin" specifies the application/entity that created the auxiliary file, and "isPublic" controls access to downloading the file. If "isPublic" is true, any user can download the file if the dataset has been published, else, access authorization is based on the access rules as defined for the DataFile itself. The "type" parameter is used to group similar auxiliary files in the UI. Currently, auxiliary files with type "DP" appear under "Differentially Private Statistics", while all other auxiliary files appear under "Other Auxiliary Files".

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export FILENAME='auxfile.json'
export FILETYPE='application/json'
export FILE_ID='12345'
export FORMAT_TAG='dpJson'
export FORMAT_VERSION='v1'
export TYPE='DP'
export SERVER_URL=https://demo.dataverse.org

curl -H X-Dataverse-key:$API_TOKEN -X POST -F "file=@$FILENAME;type=$FILETYPE" -F
↪'origin=myApp' -F 'isPublic=true' -F "type=$TYPE" "$SERVER_URL/api/access/datafile/
↪$FILE_ID/auxiliary/$FORMAT_TAG/$FORMAT_VERSION"
```

You should expect a 200 ("OK") response and JSON with information about your newly uploaded auxiliary file.

8.27.2 Downloading an Auxiliary File that Belongs to a Datafile

To download an auxiliary file, use the primary key of the datafile, and the formatTag and formatVersion (if applicable) associated with the auxiliary file. An API token is shown in the example below but it is not necessary if the auxiliary file was uploaded with isPublic=true and the dataset has been published.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export FILE_ID='12345'
export FORMAT_TAG='dpJson'
export FORMAT_VERSION='v1'

curl -H X-Dataverse-key:$API_TOKEN "$SERVER_URL/api/access/datafile/$FILE_ID/auxiliary/
↪$FORMAT_TAG/$FORMAT_VERSION"
```

8.27.3 Listing Auxiliary Files for a Datafile by Origin

To list auxiliary files, specify the primary key of the datafile (FILE_ID), and the origin associated with the auxiliary files to list (the application/entity that created them).

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export FILE_ID='12345'
export SERVER_URL=https://demo.dataverse.org
export ORIGIN='app1'

curl -H X-Dataverse-key:$API_TOKEN "$SERVER_URL/api/access/datafile/$FILE_ID/auxiliary/
↪$ORIGIN"
```

You should expect a 200 ("OK") response and a JSON array with objects representing the auxiliary files found, or a 404/Not Found response if no auxiliary files exist with that origin.

8.27.4 Deleting an Auxiliary File that Belongs to a Datafile

To delete an auxiliary file, use the primary key of the datafile, and the formatTag and formatVersion (if applicable) associated with the auxiliary file:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export FILE_ID='12345'
export FORMAT_TAG='dpJson'
export FORMAT_VERSION='v1'

curl -H X-Dataverse-key:$API_TOKEN -X DELETE "$SERVER_URL/api/access/datafile/$FILE_ID/
↪auxiliary/$FORMAT_TAG/$FORMAT_VERSION"
```

8.28 Direct DataFile Upload/Replace API

The direct Datafile Upload API is used internally to support direct upload of files to S3 storage and by tools such as the DVUploader.

Contents:

- *Overview*
- *Requesting Direct Upload of a DataFile*
- *Upload Files to S3*
- *Adding the Uploaded File to the Dataset*
- *To Add Multiple Uploaded Files to the Dataset*
- *Replacing an Existing File in the Dataset*
- *Replacing Multiple Existing Files in the Dataset*

8.28.1 Overview

Direct upload involves a series of three activities, each involving interacting with the server for a Dataverse installation:

- Requesting initiation of a transfer from the server
- Use of the pre-signed URL(s) returned in that call to perform an upload/multipart-upload of the file to S3
- A call to the server to register the file/files as part of the dataset/replace a file in the dataset or to cancel the transfer

This API is only enabled when a Dataset is configured with a data store supporting direct S3 upload. Administrators should be aware that partial transfers, where a client starts uploading the file/parts of the file and does not contact the server to complete/cancel the transfer, will result in data stored in S3 that is not referenced in the Dataverse installation (e.g. should be considered temporary and deleted.)

8.28.2 Requesting Direct Upload of a DataFile

To initiate a transfer of a file to S3, make a call to the Dataverse installation indicating the size of the file to upload. The response will include a pre-signed URL(s) that allow the client to transfer the file. Pre-signed URLs include a short-lived token authorizing the action represented by the URL.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK27U7YBV
export SIZE=1000000000

curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/datasets/:persistentId/uploadurls?
↪persistentId=$PERSISTENT_IDENTIFIER&size=$SIZE"
```

The response to this call, assuming direct uploads are enabled, will be one of two forms:

Single URL: when the file is smaller than the size at which uploads must be broken into multiple parts

```
{
  "status":"OK",
  "data":{
    "url":"...",
    "partSize":1073741824,
    "storageIdentifier":"s3://demo-dataverse-bucket:177883619b8-892ca9f7112e"
  }
}
```

Multiple URLs: when the file must be uploaded in multiple parts. The part size is set by the Dataverse installation and, for AWS-based storage, range from 5 MB to 5 GB

```
{
  "status":"OK",
  "data":{
  "urls":{
    "1":"...",
    "2":"...",
    "3":"...",
    "4":"...",
    "5":"..."
```

(continues on next page)

(continued from previous page)

```
  }
  "abort":"/api/datasets/mpupload?...",
  "complete":"/api/datasets/mpupload?..."
  "partSize":1073741824,
  "storageIdentifier":"s3://demo-dataverse-bucket:177883b000e-49cedef268ac"
}
```

The call will return a 400 (BAD REQUEST) response if the file is larger than what is allowed by the *:MaxFileUploadSizeInBytes*) and/or a quota (see *Storage Quotas for Collections*).

In the example responses above, the URLs, which are very long, have been omitted. These URLs reference the S3 server and the specific object identifier that will be used, starting with, for example, https://demo-dataverse-bucket.s3.amazonaws.com/10.5072/FK2FOQPJS/177883b000e-49cedef268ac?...

8.28.3 Upload Files to S3

The client must then use the URL(s) to PUT the file, or if the file is larger than the specified partSize, parts of the file.

In the single part case, only one call to the supplied URL is required:

```
curl -i -H 'x-amz-tagging:dv-state=temp' -X PUT -T <filename> "<supplied url>"
```

Or, if you have disabled S3 tagging (see *S3 Tagging*), you should omit the header like this:

```
curl -i -X PUT -T <filename> "<supplied url>"
```

Note that without the `-i` flag, you should not expect any output from the command above. With the `-i` flag, you should expect to see a "200 OK" response.

In the multipart case, the client must send each part and collect the 'eTag' responses from the server. The calls for this are the same as the one for the single part case except that each call should send a <partSize> slice of the total file, with the last part containing the remaining bytes. The responses from the S3 server for these calls will include the 'eTag' for the uploaded part.

To successfully conclude the multipart upload, the client must call the 'complete' URI, sending a json object including the part eTags:

```
curl -X PUT "$SERVER_URL/api/datasets/mpload?..." -d '{"1":"<eTag1 string>","2":"<eTag2␣
↪string>","3":"<eTag3 string>","4":"<eTag4 string>","5":"<eTag5 string>"}'
```

If the client is unable to complete the multipart upload, it should call the abort URL:

```
curl -X DELETE "$SERVER_URL/api/datasets/mpload?..."
```

Note

If you encounter an `HTTP 501 Not Implemented` error, ensure the `Content-Length` header is correctly set to the file or chunk size. This issue may arise when streaming files or chunks asynchronously to S3 via `PUT` requests, particularly if the library or tool you're using doesn't set the `Content-Length` header automatically.

8.28.4 Adding the Uploaded File to the Dataset

Once the file exists in the s3 bucket, a final API call is needed to add it to the Dataset. This call is the same call used to upload a file to a Dataverse installation but, rather than sending the file bytes, additional metadata is added using the "jsonData" parameter. jsonData normally includes information such as a file description, tags, provenance, whether the file is restricted, etc. For direct uploads, the jsonData object must also include values for:

- "storageIdentifier" - String, as specified in prior calls
- "fileName" - String
- "mimeType" - String
- fixity/checksum: either:
  - "md5Hash" - String with MD5 hash value, or
  - "checksum" - Json Object with "@type" field specifying the algorithm used and "@value" field with the value from that algorithm, both Strings

The allowed checksum algorithms are defined by the edu.harvard.iq.dataverse.DataFile.CheckSumType class and currently include MD5, SHA-1, SHA-256, and SHA-512

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK27U7YBV
export JSON_DATA="{'description':'My description.','directoryLabel':'data/subdir1',
↪'categories':['Data'], 'restrict':'false', 'storageIdentifier':'s3://demo-dataverse-
↪bucket:176e28068b0-1c3f80357c42', 'fileName':'file1.txt', 'mimeType':'text/plain',
↪'checksum': {'@type': 'SHA-1', '@value': '123456'}}"

curl -X POST -H "X-Dataverse-key: $API_TOKEN" "$SERVER_URL/api/datasets/:persistentId/
↪add?persistentId=$PERSISTENT_IDENTIFIER" -F "jsonData=$JSON_DATA"
```

Note that this API call can be used independently of the others, e.g. supporting use cases in which the file already exists in S3/has been uploaded via some out-of-band method. Enabling out-of-band uploads is described at *File Storage* in the Configuration Guide. With current S3 stores the object identifier must be in the correct bucket for the store, include the PID authority/identifier of the parent dataset, and be guaranteed unique, and the supplied storage identifier must be prefaced with the store identifier used in the Dataverse installation, as with the internally generated examples above.

8.28.5 To Add Multiple Uploaded Files to the Dataset

Once the files exists in the s3 bucket, a final API call is needed to add all the files to the Dataset. In this API call, additional metadata is added using the "jsonData" parameter. jsonData for this call is an array of objects that normally include information such as a file description, tags, provenance, whether the file is restricted, etc. For direct uploads, the jsonData object must also include values for:

- "description" - A description of the file
- "directoryLabel" - The "File Path" of the file, indicating which folder the file should be uploaded to within the dataset
- "storageIdentifier" - String
- "fileName" - String
- "mimeType" - String
- "fixity/checksum" either:

  - "md5Hash" - String with MD5 hash value, or
  - "checksum" - Json Object with "@type" field specifying the algorithm used and "@value" field with the value from that algorithm, both Strings

The allowed checksum algorithms are defined by the edu.harvard.iq.dataverse.DataFile.CheckSumType class and currently include MD5, SHA-1, SHA-256, and SHA-512

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV
export JSON_DATA="[{'description':'My description.','directoryLabel':'data/subdir1',
↪'categories':['Data'], 'restrict':'false', 'storageIdentifier':'s3://demo-dataverse-
↪bucket:176e28068b0-1c3f80357c42', 'fileName':'file1.txt', 'mimeType':'text/plain',
↪'checksum': {'@type': 'SHA-1', '@value': '123456'}}, \
                    {'description':'My description.','directoryLabel':'data/subdir1',
↪'categories':['Data'], 'restrict':'false', 'storageIdentifier':'s3://demo-dataverse-
↪bucket:176e28068b0-1c3f80357d53', 'fileName':'file2.txt', 'mimeType':'text/plain',
↪'checksum': {'@type': 'SHA-1', '@value': '123789'}}]"

curl -X POST -H "X-Dataverse-key: $API_TOKEN" "$SERVER_URL/api/datasets/:persistentId/
↪addFiles?persistentId=$PERSISTENT_IDENTIFIER" -F "jsonData=$JSON_DATA"
```

Note that this API call can be used independently of the others, e.g. supporting use cases in which the files already exists in S3/has been uploaded via some out-of-band method. Enabling out-of-band uploads is described at *File Storage* in the Configuration Guide. With current S3 stores the object identifier must be in the correct bucket for the store, include the PID authority/identifier of the parent dataset, and be guaranteed unique, and the supplied storage identifier must be prefaced with the store identifier used in the Dataverse installation, as with the internally generated examples above.

8.28.6 Replacing an Existing File in the Dataset

Once the file exists in the s3 bucket, a final API call is needed to register it as a replacement of an existing file. This call is the same call used to replace a file to a Dataverse installation but, rather than sending the file bytes, additional metadata is added using the "jsonData" parameter. jsonData normally includes information such as a file description, tags, provenance, whether the file is restricted, whether to allow the mimetype to change (forceReplace=true), etc. For direct uploads, the jsonData object must include values for:

- "storageIdentifier" - String, as specified in prior calls
- "fileName" - String
- "mimeType" - String
- fixity/checksum: either:
  - "md5Hash" - String with MD5 hash value, or
  - "checksum" - Json Object with "@type" field specifying the algorithm used and "@value" field with the value from that algorithm, both Strings

The allowed checksum algorithms are defined by the edu.harvard.iq.dataverse.DataFile.CheckSumType class and currently include MD5, SHA-1, SHA-256, and SHA-512. Note that the API call does not validate that the file matches the hash value supplied. If a Dataverse instance is configured to validate file fixity hashes at publication time, a mismatch would be caught at that time and cause publication to fail.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
```

(continues on next page)

(continued from previous page)

```
export FILE_IDENTIFIER=5072
export JSON_DATA='{"description":"My description.","directoryLabel":"data/subdir1",
↪"categories":["Data"], "restrict":"false", "forceReplace":"true", "storageIdentifier":
↪"s3://demo-dataverse-bucket:176e28068b0-1c3f80357c42", "fileName":"file1.txt",
↪"mimeType":"text/plain", "checksum": {"@type": "SHA-1", "@value": "123456"}}'

curl -X POST -H "X-Dataverse-key: $API_TOKEN" "$SERVER_URL/api/files/$FILE_IDENTIFIER/
↪replace" -F "jsonData=$JSON_DATA"
```

Note that this API call can be used independently of the others, e.g. supporting use cases in which the file already exists in S3/has been uploaded via some out-of-band method. Enabling out-of-band uploads is described at *File Storage* in the Configuration Guide. With current S3 stores the object identifier must be in the correct bucket for the store, include the PID authority/identifier of the parent dataset, and be guaranteed unique, and the supplied storage identifier must be prefaced with the store identifier used in the Dataverse installation, as with the internally generated examples above.

8.28.7 Replacing Multiple Existing Files in the Dataset

Once the replacement files exist in the s3 bucket, a final API call is needed to register them as replacements for existing files. In this API call, additional metadata is added using the "jsonData" parameter. jsonData for this call is array of objects that normally include information such as a file description, tags, provenance, whether the file is restricted, etc. For direct uploads, the jsonData object must include some additional values:

- "fileToReplaceId" - the id of the file being replaced
- "forceReplace" - whether to replace a file with one of a different mimetype (optional, default is false)
- "description" - A description of the file
- "directoryLabel" - The "File Path" of the file, indicating which folder the file should be uploaded to within the dataset
- "storageIdentifier" - String
- "fileName" - String
- "mimeType" - String
- "fixity/checksum" either:
  - "md5Hash" - String with MD5 hash value, or
  - "checksum" - Json Object with "@type" field specifying the algorithm used and "@value" field with the value from that algorithm, both Strings

The allowed checksum algorithms are defined by the edu.harvard.iq.dataverse.DataFile.CheckSumType class and currently include MD5, SHA-1, SHA-256, and SHA-512

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV
export JSON_DATA='[{"fileToReplaceId": 10, "description":"My description.",
↪"directoryLabel":"data/subdir1","categories":["Data"], "restrict":"false",
↪"storageIdentifier":"s3://demo-dataverse-bucket:176e28068b0-1c3f80357c42", "fileName":
↪"file1.txt", "mimeType":"text/plain", "checksum": {"@type": "SHA-1", "@value": "123456
↪"}},{"fileToReplaceId": 11, "forceReplace": true, "description":"My description.",
↪"directoryLabel":"data/subdir1","categories":["Data"], "restrict":"false",
↪"storageIdentifier":"s3://demo-dataverse-bucket:176e28068b0-1c3f80357d53", "fileName":
```

(continues on next page)

(continued from previous page)

```
↪"file2.txt", "mimeType":"text/plain", "checksum": {"@type": "SHA-1", "@value": "123789
↪"}}]'

curl -X POST -H "X-Dataverse-key: $API_TOKEN" "$SERVER_URL/api/datasets/:persistentId/
↪replaceFiles?persistentId=$PERSISTENT_IDENTIFIER" -F "jsonData=$JSON_DATA"
```

The JSON object returned as a response from this API call includes a "data" that indicates how many of the file replacements succeeded and provides per-file error messages for those that don't, e.g.

```
{
  "status": "OK",
  "data": {
    "Files": [
      {
        "storageIdentifier": "s3://demo-dataverse-bucket:176e28068b0-1c3f80357c42",
        "errorMessage": "Bad Request:The file to replace does not belong to this dataset.
↪",
        "fileDetails": {
          "fileToReplaceId": 10,
          "description": "My description.",
          "directoryLabel": "data/subdir1",
          "categories": [
            "Data"
          ],
          "restrict": "false",
          "storageIdentifier": "s3://demo-dataverse-bucket:176e28068b0-1c3f80357c42",
          "fileName": "file1.Bin",
          "mimeType": "application/octet-stream",
          "checksum": {
            "@type": "SHA-1",
            "@value": "123456"
          }
        }
      },
      {
        "storageIdentifier": "s3://demo-dataverse-bucket:176e28068b0-1c3f80357d53",
        "successMessage": "Replaced successfully in the dataset",
        "fileDetails": {
          "description": "My description.",
          "label": "file2.txt",
          "restricted": false,
          "directoryLabel": "data/subdir1",
          "categories": [
            "Data"
          ],
          "dataFile": {
            "persistentId": "",
            "pidURL": "",
            "filename": "file2.txt",
            "contentType": "text/plain",
            "filesize": 2407,
            "description": "My description.",
```

(continues on next page)

(continued from previous page)

```
            "storageIdentifier": "s3://demo-dataverse-bucket:176e28068b0-1c3f80357d53",
            "rootDataFileId": 11,
            "previousDataFileId": 11,
            "checksum": {
              "type": "SHA-1",
              "value": "123789"
            }
          }
        }
      }
    ],
    "Result": {
      "Total number of files": 2,
      "Number of files successfully replaced": 1
    }
  }
}
```

Note that this API call can be used independently of the others, e.g. supporting use cases in which the files already exists in S3/has been uploaded via some out-of-band method. Enabling out-of-band uploads is described at *File Storage* in the Configuration Guide. With current S3 stores the object identifier must be in the correct bucket for the store, include the PID authority/identifier of the parent dataset, and be guaranteed unique, and the supplied storage identifier must be prefaced with the store identifier used in the Dataverse installation, as with the internally generated examples above.

8.29 Globus Transfer API

Contents:

- *Requesting Upload or Download Parameters*
- *Performing an Upload/Transfer In*
- *Adding Files to the Dataset*
- *Downloading/Transfer Out Via Globus*

The Globus API addresses three use cases:

- Transfer to a Dataverse-managed Globus endpoint (File-based or using the Globus S3 Connector)
- Reference of files that will remain in a remote Globus endpoint
- Transfer from a Dataverse-managed Globus endpoint

The ability for Dataverse to interact with Globus endpoints is configured via a Globus store - see *Globus Storage*.

Globus transfers (or referencing a remote endpoint) for upload and download transfers involve a series of steps. These can be accomplished using the Dataverse and Globus APIs. (These are used internally by the dataverse-globus app when transfers are done via the Dataverse UI.)

8.29.1 Requesting Upload or Download Parameters

The first step in preparing for a Globus transfer/reference operation is to request the parameters relevant for a given dataset:

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/datasets/:persistentId/
↪globusUploadParameters?persistentId=$PERSISTENT_IDENTIFIER&locale=$LOCALE"
```

The response will be of the form:

```
{
        "status": "OK",
        "data": {
            "queryParameters": {
                "datasetId": 29,
                "siteUrl": "http://ec2-34-204-169-194.compute-1.amazonaws.com",
                "datasetVersion": ":draft",
                "dvLocale": "en",
                "datasetPid": "doi:10.5072/FK2/ILLPXE",
                "managed": "true",
                "fileSizeLimit": 1000000000000,
                "remainingQuota": 1000000000000,
                "endpoint": "d8c42580-6528-4605-9ad8-116a61982644"
            },
            "signedUrls": [
                {
                    "name": "requestGlobusTransferPaths",
                    "httpMethod": "POST",
                    "signedUrl": "http://ec2-34-204-169-194.compute-1.amazonaws.com/api/
↪v1/datasets/29/requestGlobusUploadPaths?until=2023-11-22T01:52:03.648&
↪user=dataverseAdmin&method=POST&
↪token=63ac4bb748d12078dded1074916508e19e6f6b61f64294d38e0b528010b07d48783cf2e975d7a1cb6d4a3c535f209b9
↪",
                    "timeOut": 300
                },
                {
                    "name": "addGlobusFiles",
                    "httpMethod": "POST",
                    "signedUrl": "http://ec2-34-204-169-194.compute-1.amazonaws.com/api/
↪v1/datasets/29/addGlobusFiles?until=2023-11-22T01:52:03.648&user=dataverseAdmin&
↪method=POST&
↪token=2aaa03f6b9f851a72e112acf584ffc0758ed0cc8d749c5a6f8c20494bb7bc13197ab123e1933f3dde2711f13b347c05
↪",
                    "timeOut": 300
                },
                {
                    "name": "getDatasetMetadata",
                    "httpMethod": "GET",
                    "signedUrl": "http://ec2-34-204-169-194.compute-1.amazonaws.com/api/
↪v1/datasets/29/versions/:draft?until=2023-11-22T01:52:03.649&user=dataverseAdmin&
↪method=GET&
↪token=1878d6a829cd5540e89c07bdaf647f1bea5314cc7a55433b0b506350dd330cad61ade3714a8ee199a7b464fb3b8cdda
↪",
                    "timeOut": 300
```

(continues on next page)

(continued from previous page)

```
                },
                {
                    "name": "getFileListing",
                    "httpMethod": "GET",
                    "signedUrl": "http://ec2-34-204-169-194.compute-1.amazonaws.com/api/
↪v1/datasets/29/versions/:draft/files?until=2023-11-22T01:52:03.650&user=dataverseAdmin&
↪method=GET&
↪token=78e8ca8321624f42602af659227998374ef3788d0feb43d696a0e19086e0f2b3b66b96981903a1565e836416c504b624
↪",
                    "timeOut": 300
                }
            ]
        }
  }
```

The response includes the id for the Globus endpoint to use along with several parameters and signed URLs. The parameters include whether the Globus endpoint is "managed" by Dataverse and, if so, if there is a "fileSizeLimit" (see *:MaxFileUploadSizeInBytes*) that will be enforced and/or, if there is a quota (see *Storage Quotas for Collections*) on the overall size of data that can be upload, what the "remainingQuota" is. Both are in bytes.

Note that while Dataverse will not add files that violate the size or quota rules, Globus itself doesn't enforce these during the transfer. API users should thus check the size of the files they intend to transfer before submitting a transfer request to Globus.

The getDatasetMetadata and getFileListing URLs are just signed versions of the standard Dataset metadata and file listing API calls. The other two are Globus specific.

If called for a dataset using a store that is configured with a remote Globus endpoint(s), the return response is similar but the response includes a the "managed" parameter will be false, the "endpoint" parameter is replaced with a JSON array of "referenceEndpointsWithPaths" and the requestGlobusTransferPaths and addGlobusFiles URLs are replaced with ones for requestGlobusReferencePaths and addFiles. All of these calls are described further below.

The call to set up for a transfer out (download) is similar:

```
curl -H "X-Dataverse-key:$API_TOKEN" "$SERVER_URL/api/datasets/:persistentId/
↪globusDownloadParameters?persistentId=$PERSISTENT_IDENTIFIER&locale=$LOCALE"
```

Note that this API call supports an additional downloadId query parameter. This is only used when the globus-dataverse app is called from the Dataverse user interface. There is no need to use it when calling the API directly.

The returned response includes the same getDatasetMetadata and getFileListing URLs as in the upload case and includes "monitorGlobusDownload" and "requestGlobusDownload" URLs. The response will also indicate whether the store is "managed" and will provide the "endpoint" from which downloads can be made.

8.29.2 Performing an Upload/Transfer In

The information from the API call above can be used to provide a user with information about the dataset and to prepare to transfer (managed=true) or to reference files (managed=false).

Once the user identifies which files are to be added, the requestGlobusTransferPaths or requestGlobusReferencePaths URLs can be called. These both reference the same API call but must be used with different entries in the JSON body sent:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV
export LOCALE=en-US
export JSON_DATA="... (SEE BELOW)"

curl -H "X-Dataverse-key:$API_TOKEN" -H "Content-type:application/json" -X POST  -d "
↪$JSON_DATA" "$SERVER_URL/api/datasets/:persistentId/requestGlobusUploadPaths?
↪persistentId=$PERSISTENT_IDENTIFIER"
```

Note that when using the dataverse-globus app or the return from the previous call, the URL for this call will be signed and no API_TOKEN is needed.

In the managed case, the JSON body sent must include the id of the Globus user that will perform the transfer and the number of files that will be transferred:

```
{
  "principal":"d15d4244-fc10-47f3-a790-85bdb6db9a75",
  "numberOfFiles":2
}
```

In the remote reference case, the JSON body sent must include the Globus endpoint/paths that will be referenced:

```
{
  "referencedFiles":[
    "d8c42580-6528-4605-9ad8-116a61982644/hdc1/test1.txt"
  ]
}
```

The response will include a JSON object. In the managed case, the map is from new assigned file storageidentifiers and specific paths on the managed Globus endpoint:

```
{
  "status":"OK",
  "data":{
    "globusm://18b49d3688c-62137dcb06e4":"/hdc1/10.5072/FK2/ILLPXE/18b49d3688c-
↪62137dcb06e4",
    "globusm://18b49d3688c-5c17d575e820":"/hdc1/10.5072/FK2/ILLPXE/18b49d3688c-
↪5c17d575e820"
  }
}
```

In the managed case, the specified Globus principal is granted write permission to the specified endpoint/path, which will allow initiation of a transfer from the external endpoint to the managed endpoint using the Globus API. The permission will be revoked if the transfer is not started and the next call to Dataverse to finish the transfer are not made within a short time (configurable, default of 5 minutes).

In the remote/reference case, the map is from the initially supplied endpoint/paths to the new assigned file storageidentifiers:

```
{
  "status":"OK",
  "data":{
    "d8c42580-6528-4605-9ad8-116a61982644/hdc1/test1.txt":"globus://18bf8c933f4-
```

(continues on next page)

(continued from previous page)

```
↪ed2661e7d19b//d8c42580-6528-4605-9ad8-116a61982644/hdc1/test1.txt"
  }
}
```

8.29.3 Adding Files to the Dataset

In the managed case, you must initiate a Globus transfer and take note of its task identifier. As in the JSON example below, you will pass it as `taskIdentifier` along with details about the files you are transferring:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV
export JSON_DATA='{"taskIdentifier":"3f530302-6c48-11ee-8428-378be0d9c521", \
                  "files": [{"description":"My description.","directoryLabel":"data/
↪subdir1","categories":["Data"], "restrict":"false", "storageIdentifier":"globusm://
↪18b3972213f-f6b5c2221423", "fileName":"file1.txt", "mimeType":"text/plain", "checksum
↪": {"@type": "MD5", "@value": "1234"}}, \
                  {"description":"My description.","directoryLabel":"data/subdir1",
↪"categories":["Data"], "restrict":"false", "storageIdentifier":"globusm://18b39722140-
↪50eb7d3c5ece", "fileName":"file2.txt", "mimeType":"text/plain", "checksum": {"@type":
↪"MD5", "@value": "2345"}}]}'

curl -H "X-Dataverse-key:$API_TOKEN" -H "Content-type:multipart/form-data" -X POST "
↪$SERVER_URL/api/datasets/:persistentId/addGlobusFiles?persistentId=$PERSISTENT_
↪IDENTIFIER" -F "jsonData=$JSON_DATA"
```

Note that the mimetype is multipart/form-data, matching the /addFiles API call. Also note that the API_TOKEN is not needed when using a signed URL.

With this information, Dataverse will begin to monitor the transfer and when it completes, will add all files for which the transfer succeeded. As the transfer can take significant time and the API call is asynchronous, the only way to determine if the transfer succeeded via API is to use the standard calls to check the dataset lock state and contents.

Once the transfer completes, Dataverse will remove the write permission for the principal.

An alternative, experimental implementation of Globus polling of ongoing upload transfers has been added in v6.4. This new framework does not rely on the instance staying up continuously for the duration of the transfer and saves the state information about Globus upload requests in the database. Due to its experimental nature it is not enabled by default. See the *dataverse.feature.globus-use-experimental-async-framework* feature flag and the JVM option *dataverse.files.globus-monitoring-server*.

Note that when using a managed endpoint that uses the Globus S3 Connector, the checksum should be correct as Dataverse can validate it. For file-based endpoints, the checksum should be included if available but Dataverse cannot verify it.

In the remote/reference case, where there is no transfer to monitor, the standard /addFiles API call (see *Adding the Uploaded File to the Dataset*) is used instead. There are no changes for the Globus case.

8.29.4 Downloading/Transfer Out Via Globus

To begin downloading files, the requestGlobusDownload URL is used:

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV

curl -H "X-Dataverse-key:$API_TOKEN" -H "Content-type:application/json" -X POST -d "
↪$JSON_DATA" "$SERVER_URL/api/datasets/:persistentId/requestGlobusDownload?persistentId=
↪$PERSISTENT_IDENTIFIER"
```

The JSON body sent should include a list of file ids to download and, for a managed endpoint, the Globus principal that will make the transfer:

```
export JSON_DATA='{ \
  "principal":"d15d4244-fc10-47f3-a790-85bdb6db9a75", \
  "fileIds":[60, 61] \
}'
```

Note that this API call takes an optional downloadId parameter that is used with the dataverse-globus app. When downloadId is included, the list of fileIds is not needed.

The response is a JSON object mapping the requested file Ids to Globus endpoint/paths. In the managed case, the principal will have been given read permissions for the specified paths:

```
{
  "status":"OK",
  "data":{
    "60": "d8c42580-6528-4605-9ad8-116a61982644/hdc1/10.5072/FK2/ILLPXE/18bf3af9c78-
↪92b8e168090e",
   "61": "d8c42580-6528-4605-9ad8-116a61982644/hdc1/10.5072/FK2/ILLPXE/18bf3af9c78-
↪c8d81569305c"
  }
}
```

For the remote case, the use can perform the transfer without further contact with Dataverse. In the managed case, the user must initiate the transfer via the Globus API and then inform Dataverse. Dataverse will then monitor the transfer and revoke the read permission when the transfer is complete. (Not making this last call could result in failure of the transfer.)

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export PERSISTENT_IDENTIFIER=doi:10.5072/FK2/7U7YBV

curl -H "X-Dataverse-key:$API_TOKEN" -H "Content-type:application/json" -X POST -d "
↪$JSON_DATA" "$SERVER_URL/api/datasets/:persistentId/monitorGlobusDownload?persistentId=
↪$PERSISTENT_IDENTIFIER"
```

The JSON body sent just contains the task identifier for the transfer:

```
export JSON_DATA='{ \
  "taskIdentifier":"b5fd01aa-8963-11ee-83ae-d5484943e99a" \
}'
```

8.30 Dataset Semantic Metadata API

Contents:

- *Get Dataset Metadata*
- *Add Dataset Metadata*
- *Delete Dataset Metadata*
- *Create a Dataset*
- *Create a Dataset with a Dataset Type*

The OAI_ORE metadata export format represents Dataset metadata using json-ld (see the *Metadata Export* section). As part of an RDA-supported effort to allow import of Datasets exported as Bags with an included OAI_ORE metadata file, an experimental API has been created that provides a json-ld alternative to the v1.0 API calls to get/set/delete Dataset metadata in the *Native API*.

You may prefer to work with this API if you are building a tool to import from a Bag/OAI-ORE source or already work with json-ld representations of metadata, or if you prefer the flatter json-ld representation to Dataverse software's json representation (which includes structure related to the metadata blocks involved and the type/multiplicity of the metadata fields.) You may not want to use this API if you need stability and backward compatibility (the 'experimental' designation for this API implies that community feedback is desired and that, in future Dataverse software versions, the API may be modified based on that feedback).

Note: The examples use the 'application/ld+json' mimetype. For compatibility reasons, the APIs also be used with mimetype "application/json-ld"

8.30.1 Get Dataset Metadata

To get the json-ld formatted metadata for a Dataset, specify the Dataset ID (DATASET_ID) or Persistent identifier (DATASET_PID), and, for specific versions, the version number.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export DATASET_ID='12345'
export DATASET_PID='doi:10.5072/FK2A1B2C3'
export VERSION='1.0'
export SERVER_URL=https://demo.dataverse.org

Example 1: Get metadata for version '1.0'

  curl -H X-Dataverse-key:$API_TOKEN -H 'Accept: application/ld+json' "$SERVER_URL/api/
↪datasets/$DATASET_ID/versions/$VERSION/metadata"

Example 2: Get metadata for the latest version using the DATASET PID

  curl -H X-Dataverse-key:$API_TOKEN -H 'Accept: application/ld+json' "$SERVER_URL/api/
↪datasets/:persistentId/metadata?persistentId=$DATASET_PID"
```

You should expect a 200 ("OK") response and JSON-LD mirroring the OAI-ORE representation in the returned 'data' object.

8.30.2 Add Dataset Metadata

To add json-ld formatted metadata for a Dataset, specify the Dataset ID (DATASET_ID) or Persistent identifier (DATASET_PID). Adding '?replace=true' will overwrite an existing metadata value. The default (replace=false) will only add new metadata or add a new value to a multi-valued field.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export DATASET_ID='12345'
export DATASET_PID='doi:10.5072/FK2A1B2C3'
export VERSION='1.0'
export SERVER_URL=https://demo.dataverse.org

Example: Change the Dataset title

  curl -X PUT -H X-Dataverse-key:$API_TOKEN -H 'Content-Type: application/ld+json' -d '{
↪"title": "Submit menu test", "@context":{"title": "http://purl.org/dc/terms/title"}}' "
↪$SERVER_URL/api/datasets/$DATASET_ID/metadata?replace=true"

Example 2: Add a description using the DATASET PID

  curl -X PUT -H X-Dataverse-key:$API_TOKEN -H 'Content-Type: application/ld+json' -d '{
↪"citation:dsDescription": {"citation:dsDescriptionValue": "New description"}, "@context
↪":{"citation": "https://dataverse.org/schema/citation/"}}' "$SERVER_URL/api/datasets/
↪:persistentId/metadata?persistentId=$DATASET_PID"
```

You should expect a 200 ("OK") response indicating whether a draft Dataset version was created or an existing draft was updated.

8.30.3 Delete Dataset Metadata

To delete metadata for a Dataset, send a json-ld representation of the fields to delete and specify the Dataset ID (DATASET_ID) or Persistent identifier (DATASET_PID).

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export DATASET_ID='12345'
export DATASET_PID='doi:10.5072/FK2A1B2C3'
export VERSION='1.0'
export SERVER_URL=https://demo.dataverse.org

Example: Delete the TermsOfUseAndAccess 'restrictions' value 'No restrictions' for the␣
↪latest version using the DATASET PID

  curl -X PUT -H X-Dataverse-key:$API_TOKEN -H 'Content-Type: application/ld+json' -d '{
↪"https://dataverse.org/schema/core#restrictions":"No restrictions"}' "$SERVER_URL/api/
↪datasets/:persistentId/metadata/delete?persistentId=$DATASET_PID"
```

Note, this example uses the term URI directly rather than adding an `@context` element. You can use either form in any of these API calls.

You should expect a 200 ("OK") response indicating whether a draft Dataset version was created or an existing draft was updated.

8.30.4 Create a Dataset

Specifying the Content-Type as application/ld+json with the existing /api/dataverses/{id}/datasets API call (see *Create a Dataset in a Dataverse Collection*) supports using the same metadata format when creating a Dataset.

With curl, this is done by adding the following header:

```
-H 'Content-Type: application/ld+json'

.. code-block:: bash

export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export DATAVERSE_ID=root
export PERSISTENT_IDENTIFIER=doi:10.5072/FK27U7YBV

curl -H X-Dataverse-key:$API_TOKEN -H 'Content-Type: application/ld+json' -X POST
↪$SERVER_URL/api/dataverses/$DATAVERSE_ID/datasets --upload-file dataset-create.jsonld
```

An example jsonld file is available at `dataset-create.jsonld` (`dataset-create_en.jsonld` is a version that sets the metadata language (see *:MetadataLanguages*) to English (en).)

8.30.5 Create a Dataset with a Dataset Type

By default, datasets are given the type "dataset" but if your installation had added additional types (see *Add Dataset Type*), you can specify the type.

An example JSON-LD file is available at `dataset-create-software.jsonld`.

You can use this file with the normal *Create a Dataset* endpoint above.

See also *Dataset Types*.

8.31 Dataset Migration API

The Dataverse software includes several ways to add Datasets originally created elsewhere (not to mention Harvesting capabilities). These include the Sword API (see the *SWORD API* guide) and the /dataverses/{id}/datasets/:import methods (json and ddi) (see the *Native API* guide).

This experimental migration API offers an additional option with some potential advantages:

- Metadata can be specified using the json-ld format used in the OAI-ORE metadata export. Please note that the json-ld generated by OAI-ORE metadata export is not directly compatible with the Migration API. OAI-ORE export nests resource metadata under `ore:describes` wrapper and Dataset Migration API requires that metadata is on the root level. Please check example file below for reference.
  - If you need a tool to convert OAI-ORE exported json-ld into a format compatible with the Dataset Migration API, or if you need to generate compatible json-ld from sources other than an existing Dataverse installation, the BaseX database engine, used together with the XQuery language, provides an efficient solution. Please see example script `transform-oai-ore-jsonld.xq` for a simple conversion from exported OAI-ORE json-ld to a Dataset Migration API -compatible version.
- Existing publication dates and PIDs are maintained (currently limited to the case where the PID can be managed by the Dataverse software, e.g. where the authority and shoulder match those the software is configured for)
- Updating the PID at the provider can be done immediately or later (with other existing APIs).

- Adding files can be done via the standard APIs, including using direct-upload to S3.

This API consists of 2 calls: one to create an initial Dataset version, and one to 'republish' the dataset through Dataverse with a specified publication date. Both calls require super-admin privileges.

These calls can be used in concert with other API calls to add files, update metadata, etc. before the 'republish' step is done.

8.31.1 Start Migrating a Dataset into a Dataverse Collection

> **Note**
>
> This action requires a Dataverse installation account with superuser permissions.

To import a dataset with an existing persistent identifier (PID), the provided json-ld metadata should include it.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export DATAVERSE_ID=root

curl -H X-Dataverse-key:$API_TOKEN -X POST $SERVER_URL/api/dataverses/$DATAVERSE_ID/
↪datasets/:startmigration --upload-file dataset-migrate.jsonld
```

An example jsonld file is available at `dataset-migrate.jsonld` . Note that you would need to replace the PID in the sample file with one supported in your Dataverse instance.

You also need to replace the `dataverse.siteUrl` in the json-ld `@context` with your current Dataverse site URL. This is necessary to define a local URI for metadata terms originating from community metadata blocks (in the case of the example file, from the Social Sciences and Humanities and Geospatial blocks).

Currently, as of Dataverse 6.5 and earlier, community metadata blocks do not assign a default global URI to the terms used in the block in contrast to citation metadata, which has global URI defined.

8.31.2 Publish a Migrated Dataset

The call above creates a Dataset. Once it is created, other APIs can be used to add files, add additional metadata, etc. When a version is complete, the following call can be used to publish it with its original publication date.

> **Note**
>
> This action requires a Dataverse installation account with superuser permissions.

```
export API_TOKEN=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org

curl -H 'Content-Type: application/ld+json' -H X-Dataverse-key:$API_TOKEN -X POST -d '{
↪"schema:datePublished": "2020-10-26","@context":{ "schema":"http://schema.org/"}}' "
↪$SERVER_URL/api/datasets/{id}/actions/:releasemigrated"
```

datePublished is the only metadata supported in this call.

An optional query parameter: updatepidatprovider (default is false) can be set to true to automatically update the metadata and targetUrl of the PID at the provider. With this set true, the result of this call will be that the PID redirects to this dataset rather than the dataset in the source repository.

```
curl -H 'Content-Type: application/ld+json' -H X-Dataverse-key:$API_TOKEN -X POST -d '{
↪"schema:datePublished": "2020-10-26","@context":{ "schema":"http://schema.org/"}}' "
↪$SERVER_URL/api/datasets/{id}/actions/:releasemigrated?updatepidatprovider=true"
```

If the parameter is not added and set to true, other existing APIs can be used to update the PID at the provider later, e.g. *Update Target URL for a Published Dataset at the PID provider*

8.32 Workflows

The Dataverse Software has a flexible workflow mechanism that can be used to trigger actions before and after Dataset publication.

Contents:

- *Introduction*
  - *Administration*
  - *Available Steps*
    - *log*
    - *pause*
    - *pause/message*
    - *http/sr*
    - *http/authext*
    - *archiver*
    - *coarNotifyRelationshipAnnouncement*

8.32.1 Introduction

The Dataverse Software can perform two sequences of actions when datasets are published: one prior to publishing (marked by a `PrePublishDataset` trigger), and one after the publication has succeeded (`PostPublishDataset`). The pre-publish workflow is useful for having an external system prepare a dataset for being publicly accessed (a possibly lengthy activity that requires moving files around, uploading videos to a streaming server, etc.), or to start an approval process. A post-publish workflow might be used for sending notifications about the newly published dataset.

Workflow steps are created using *step providers*. The Dataverse Software ships with an internal step provider that offers some basic functionality, and with the ability to load 3rd party step providers (currently disabled). This allows installations to implement functionality they need without changing the Dataverse Software source code.

Steps can be internal (say, writing some data to the log) or external. External steps involve the Dataverse Software sending a request to an external system, and waiting for the system to reply. The wait period is arbitrary, and so allows the external system unbounded operation time. This is useful, e.g., for steps that require human intervention, such as manual approval of a dataset publication.

The external system reports the step result back to the Dataverse installation, by sending a HTTP `POST` command to `api/workflows/{invocation-id}` with Content-Type: text/plain. The body of the request is passed to the paused step for further processing.

Steps can define messages to send to the log and to users. If defined, the message to users is sent as a user notification (creating an email and showing in the user notification tab) and will show once for the given user if/when they view the relevant dataset page. The latter provides a means for the asynchronous workflow execution to report success or failure analogous to the way the publication and other processes report on the page.

If a step in a workflow fails, the Dataverse installation makes an effort to roll back all the steps that preceded it. Some actions, such as writing to the log, cannot be rolled back. If such an action has a public external effect (e.g. send an EMail to a mailing list) it is advisable to put it in the post-release workflow.

Tip

For invoking external systems using a REST api, the Dataverse Software's internal step provider offers two steps for sending and receiving customizable HTTP requests. *http/sr* and *http/authExt*, detailed below, with the latter able to use the API to make changes to the dataset being processed. (Both lock the dataset to prevent other processes from changing the dataset between the time the step is launched to when the external process responds to the Dataverse instance.)

Administration

A Dataverse installation stores a set of workflows in its database. Workflows can be managed using the `api/admin/workflows/` endpoints of the *Native API*. Sample workflow files are available in `scripts/api/data/workflows`.

At the moment, defining a workflow for each trigger is done for the entire instance, using the endpoint `api/admin/workflows/default/«trigger type»`.

In order to prevent unauthorized resuming of workflows, the Dataverse installation maintains a "white list" of IP addresses from which resume requests are honored. This list is maintained using the `/api/admin/workflows/ip-whitelist` endpoint of the *Native API*. By default, the Dataverse installation honors resume requests from localhost only (`127.0.0.1;::1`), so set-ups that use a single server work with no additional configuration.

Note: these settings are also exposed and manageable via the Settings API. See *:WorkflowsAdminIpWhitelist*, *:PrePublishDatasetWorkflowId* and *:PostPublishDatasetWorkflowId*

Available Steps

The Dataverse Software has an internal step provider, whose id is `:internal`. It offers the following steps:

log

A step that writes data about the current workflow invocation to the instance log. It also writes the messages in its `parameters` map.

```
{
   "provider":":internal",
   "stepType":"log",
   "parameters": {
       "aMessage": "message content",
       "anotherMessage": "message content, too"
```

(continues on next page)

(continued from previous page)

```
    }
}
```

pause

A step that pauses the workflow. The workflow is paused until a POST request is sent to `/api/workflows/{invocation-id}`. Sending 'fail' in the POST body (Content-type:text/plain) will trigger a failure and workflow rollback. All other responses are considered as successes. The pause step is intended for testing - the invocationId required to end the pause is only available in the log (and database). Adding a parameter (see log step) with the key/value "authorized":"true" will allow the invocationId to be used as a credential as with the http/authext step below.

```
{
    "provider":":internal",
    "stepType":"pause"
}
```

pause/message

A variant of the pause step that pauses the workflow and allows the external process to send a success/failure message. The workflow is paused until a POST request is sent to `/api/workflows/{invocation-id}`. The response in the POST body (Content-type:application/json) should be a json object (the same as for the http/extauth step) containing: - "status" - can be "success" or "failure" - "reason" - a message that will be logged - "message" - a message to send to the user that will be sent as a notification and as a banner on the relevant dataset page. An unparsable reponse will be considered a Failure that will be logged with no user message. (See the http/authext step for an example POST call)

```
{
    "provider":":internal",
    "stepType":"pause/message"
}
```

http/sr

A step that sends a HTTP request to an external system, and then waits for a response. The response has to match a regular expression specified in the step parameters. The url, content type, and message body can use data from the workflow context, using a simple markup language. This step has specific parameters for rollback. The workflow is restarted when the external system replies with a POST request to `/api/workflows/{invocation-id}`. Responses starting with "OK" (Content-type:text/plain) are considered successes. Other responses will be considered failures and trigger workflow rollback.

```
{
  "provider":":internal",
  "stepType":"http/sr",
  "parameters": {
      "url":"http://localhost:5050/dump/${invocationId}",
      "method":"POST",
      "contentType":"text/plain",
      "body":"START RELEASE ${dataset.id} as ${dataset.displayName}",
      "expectedResponse":"OK.*",
```

(continues on next page)

(continued from previous page)

```
      "rollbackUrl":"http://localhost:5050/dump/${invocationId}",
      "rollbackMethod":"DELETE ${dataset.id}"
  }
}
```

Available variables are:

- invocationId
- dataset.id
- dataset.identifier
- dataset.globalId
- dataset.displayName
- dataset.citation
- minorVersion
- majorVersion
- releaseStatus

http/authext

Similar to the *http/sr* step. A step that sends a HTTP request to an external system, and then waits for a response. The receiver can use the invocationId of the workflow in lieu of an api key to perform work on behalf of the user launching the workflow. The invocationId must be sent as an 'X-Dataverse-invocationId' HTTP Header or as an ?invocationId= query parameter. *Note that any external process started using this step then has the ability to access a Dataverse instance via the API as the user.* Once this step completes and responds, the invocationId is invalidated and will not allow further access.

The url, content type, and message body can use data from the workflow context, using a simple markup language. This step has specific parameters for rollback. The workflow is restarted when the external system replies with a POST request to `/api/workflows/{invocation-id}` (Content-Type: application/json).

The response has is expected to be a json object with three keys: - "status" - can be "success" or "failure" - "reason" - a message that will be logged - "message" - a message to send to the user that will be sent as a notification and as a banner on the relevant dataset page.

```
export INVOCATION_ID=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
export SERVER_URL=https://demo.dataverse.org
export MESSAGE={"status":"success", "reason":"Workflow completed in 10 seconds", "message
↪":"An external workflow to virus check your data was successfully run prior to␣
↪publication of your data"}

curl -H 'Content-Type:application/json' -X POST -d $MESSAGE "$SERVER_URL/api/workflows/
↪$INVOCATION_ID"
```

```
{
  "provider":":internal",
  "stepType":"http/authext",
  "parameters": {
      "url":"http://localhost:5050/dump/${invocationId}",
```

(continues on next page)

(continued from previous page)

```
      "method":"POST",
      "contentType":"text/plain",
      "body":"START RELEASE ${dataset.id} as ${dataset.displayName}",
      "rollbackUrl":"http://localhost:5050/dump/${invocationId}",
      "rollbackMethod":"DELETE ${dataset.id}"
  }
}
```

Available variables are:

- invocationId
- dataset.id
- dataset.identifier
- dataset.globalId
- dataset.displayName
- dataset.citation
- minorVersion
- majorVersion
- releaseStatus

archiver

A step that sends an archival copy of a Dataset Version to a configured archiver, e.g. the DuraCloud interface of Chronopolis. See *RDA BagIt (BagPack) Archiving* for further detail.

Note - the example step includes two settings required for any archiver, three (DuraCloud*) that are specific to DuraCloud, and the optional BagGeneratorThreads setting that controls parallelism when creating the Bag.

```
{
  "provider":":internal",
  "stepType":"archiver",
  "parameters": {
    "stepName":"archive submission"
  },
  "requiredSettings": {
    ":ArchiverClassName": "string",
    ":ArchiverSettings": "string",
    ":DuraCloudHost":"string",
    ":DuraCloudPort":"string",
    ":DuraCloudContext":"string",
    ":BagGeneratorThreads":"string"
  }
}
```

coarNotifyRelationshipAnnouncement

A step that sends a COAR Notify Relationship Announcement message, using the Linked Data Notification (LDN) message standard, to a specific set of LDN Inboxes announcing a relationship between a newly published/available dataset and an external resource (e.g. one managed by the recipient).

The two parameters are * `:COARNotifyRelationshipAnnouncementTriggerFields` - a list of metadata field types that can trigger messages. Separate messages will be sent for each field (whether of the same type or not). * `:COARNotifyRelationshipAnnouncementTargets` - a JSON Array of JSON objects containing `id`, `inbox`, and `type` fields as required by the COAR Notify Relationship Announcement specification . The `inbox` value should be the full URL of the target LDN inbox to which messages should be sent, e.g. `{"id": "https://dashv7-dev.lib.harvard.edu","inbox": "https://dashv7-api-dev.lib.harvard.edu/server/ldn/inbox","type": ["Service"]}`).

```
{
  "provider":":internal",
  "stepType":"coarNotifyRelationshipAnnouncement",
  "parameters": {
    "stepName":"COAR Notify Relationship Announcement"
  },
  "requiredSettings": {
    ":COARNotifyRelationshipAnnouncementTriggerFields": "string",
    ":COARNotifyRelationshipAnnouncementTargets": "string"
  }
}
```

8.33 Font Custom

As mentioned under *Font Custom Icon Font* in the Style Guide, Dataverse uses Font Custom to create custom icon fonts.

Contents:

- *Previewing Icons*
- *Install Font Custom*
  - *Ruby Version*
  - *Install Dependencies and Font Custom Gem*
- *Updating Icons*
  - *Updating the Guides Icons*
  - *Updating the App Icons*
  - *Committing Changes to Git*
- *Caveats About Font Custom*

8.33.1 Previewing Icons

The process below updates a preview page that you can use to see how the icons look.

In scripts/icons/svg in the source tree, you can see the SVG files that the icons are built from.

8.33.2 Install Font Custom

You'll need Font Custom and its dependencies installed if you want to update the icons.

Ruby Version

Font Custom is written in Ruby. Ruby 3.0 didn't "just work" with FontAwesome but Ruby 2.7 was fine.

RVM is a good way to install a specific version of Ruby: https://rvm.io

Install Dependencies and Font Custom Gem

The brew commands below assume you are on a Mac.

```
brew tap bramstein/webfonttools
brew update
brew install woff2
brew install sfnt2woff
brew install fontforge
brew install eot-utils
gem install fontcustom
```

(`brew install sfnt2woff` isn't currently listed in the FontCustom README but it's in mentioned in https://github.com/FontCustom/fontcustom/pull/385)

If `fontcustom --help` works now, you have it installed.

8.33.3 Updating Icons

Navigate to `scripts/icons` in the source tree (or online) and you will find:

- An `svg` directory containing the "source" for the icons.
- Scripts to update the icons.

There is a copy of these icons in both the app and the guides. We'll update the guides first because it's much quicker to iterate and notice any problems with the icons.

Updating the Guides Icons

Run `docs.sh` and then open `../../doc/sphinx-guides/source/_static/fontcustom-preview.html` in a browser to look at the icons. (This is the preview page mentioned above that gets incorporated in the next Sphinx build.)

Update any files in the `svg` directory and run the script again to see any differences.

Note that Font Custom creates font files with unique names. For this reason, we should remove the old files from git as we add the new ones. The script deletes the old files for you but in a step below we'll do a `git add` to stage this change.

Updating the App Icons

Assuming you're happy with how the icons look in the preview page in the guides, you can move on to updating the icons in the Dataverse app itself.

This time the script is called `app.sh` and it works the same way with the addition of tweaking some URLs. Go ahead and run this script and do a full "clean and build" before making sure the changes are visible in the application.

Committing Changes to Git

As mentioned above, icons are in both the app and the docs. Again, because the filenames change, we should make sure the old files are removed from git.

From the root of the repo, run the following:

```
git add doc/sphinx-guides/source/_static
git add src/main/webapp/resources
```

That should be enough to make sure old files are replaced by new ones. At this point, you can commit and make a pull request.

8.33.4 Caveats About Font Custom

Font Custom is a useful tool and has an order of magnitude more stars on GitHub than its competitors. However, an issue suggests that the tool is somewhat abandoned. Its domain has expired but you can still get at what used to be its website at https://fontcustom.github.io/fontcustom/

8.34 Classic Dev Environment

These are the old instructions we used for Dataverse 4 and 5. They should still work but these days we favor running Dataverse in Docker as described in *Development Environment*.

These instructions are purposefully opinionated and terse to help you get your development environment up and running as quickly as possible! Please note that familiarity with running commands from the terminal is assumed.

Contents:

- *Quick Start (Docker)*

- *Classic Dev Environment*
- *Set Up Dependencies*
  - *Supported Operating Systems*
  - *Install Java*
  - *Install Netbeans or Maven*
  - *Install Homebrew (Mac Only)*
  - *Clone the Dataverse Software Git Repo*
  - *Build the Dataverse Software War File*
  - *Install jq*
  - *Install Payara*
  - *Install Service Dependencies Directly on localhost*
    - *Install PostgreSQL*
    - *Install Solr*
  - *Install Service Dependencies Using Docker Compose*
- *Run the Dataverse Software Installer Script*
  - *Verify the Dataverse Software is Running*
- *Configure Your Development Environment for Publishing*
- *Configure Your Development Environment for GUI Edits*
- *Next Steps*

8.34.1 Quick Start (Docker)

The quickest way to get Dataverse running is in Docker as explained in *Development Usage* section of the Container Guide.

8.34.2 Classic Dev Environment

Since before Docker existed, we have encouraged installing Dataverse and all its dependencies directly on your development machine, as described below. This can be thought of as the "classic" development environment for Dataverse.

However, in 2023 we decided that we'd like to encourage all developers to start using Docker instead and opened https://github.com/IQSS/dataverse/issues/9616 to indicate that we plan to rewrite this page to recommend the use of Docker.

There's nothing wrong with the classic instructions below and we don't plan to simply delete them. They are a valid alternative to running Dataverse in Docker. We will likely move them to another page.

8.34.3 Set Up Dependencies

Supported Operating Systems

Mac OS X or Linux is required because the setup scripts assume the presence of standard Unix utilities.

Windows is gaining support through Docker as described in the *Windows Development* section.

Install Java

The Dataverse Software requires Java 21.

We suggest downloading OpenJDK from https://adoptopenjdk.net

On Linux, you are welcome to use the OpenJDK available from package managers.

Install Netbeans or Maven

NetBeans IDE is recommended, and can be downloaded from https://netbeans.org . Developers may use any editor or IDE. We recommend NetBeans because it is free, works cross platform, has good support for Jakarta EE projects, and includes a required build tool, Maven.

Below we describe how to build the Dataverse Software war file with Netbeans but if you prefer to use only Maven, you can find installation instructions in the *Tools* section.

Install Homebrew (Mac Only)

On Mac, install Homebrew to simplify the steps below: https://brew.sh

Clone the Dataverse Software Git Repo

Fork https://github.com/IQSS/dataverse and then clone your fork like this:

```
git clone git@github.com:[YOUR GITHUB USERNAME]/dataverse.git
```

Build the Dataverse Software War File

If you installed Netbeans, follow these steps:

- Launch Netbeans and click "File" and then "Open Project". Navigate to where you put the Dataverse Software code and double-click "Dataverse" to open the project.
- If you see "resolve project problems," go ahead and let Netbeans try to resolve them. This will probably including downloading dependencies, which can take a while.
- Allow Netbeans to install nb-javac (required for Java 8 and below).
- Select "Dataverse" under Projects and click "Run" in the menu and then "Build Project (Dataverse)". Check back for "BUILD SUCCESS" at the end.

If you installed Maven instead of Netbeans, run `mvn package`. Check for "BUILD SUCCESS" at the end.

NOTE: Do you use a locale different than `en_US.UTF-8` on your development machine? Are you in a different timezone than Harvard (Eastern Time)? You might experience issues while running tests that were written with these settings in mind. The Maven `pom.xml` tries to handle this for you by setting the locale to `en_US.UTF-8` and timezone `UTC`, but more, not yet discovered building or testing problems might lurk in the shadows.

Install jq

On Mac, run this command:

```
brew install jq
```

On Linux, install `jq` from your package manager or download a binary from https://stedolan.github.io/jq/

Install Payara

See *Payara* under the Installation Guide for how to download the correct version of Payara. For the instructions below, we'll assume you've downloaded it to `/tmp` and will refer to it as `payara*.zip` (so we don't have to maintain the version number in multiple places).

To install Payara, run the following commands:

```
cd /usr/local
```

```
cp /tmp/payara*zip .
```

```
sudo unzip payara*.zip
```

```
sudo chown -R $USER /usr/local/payara7
```

If nexus.payara.fish is ever down for maintenance, Payara distributions are also available from https://repo1.maven.org/maven2/fish/payara/distributions/payara/

Install Service Dependencies Directly on localhost

Install PostgreSQL

See `docker-compose-dev.yml` for the version of PostgreSQL to use. Below we use "PGVERSION" to mean the version of PostgreSQL.

On Mac, go to https://www.postgresql.org/download/macosx/ and choose "Interactive installer by EDB" option. When prompted to set a password for the "database superuser (postgres)" just enter "password".

After installation is complete, make a backup of the `pg_hba.conf` file like this:

```
sudo cp /Library/PostgreSQL/PGVERSION/data/pg_hba.conf /Library/PostgreSQL/PGVERSION/
data/pg_hba.conf.orig
```

Then edit `pg_hba.conf` with an editor such as vi:

```
sudo vi /Library/PostgreSQL/PGVERSION/data/pg_hba.conf
```

In the "METHOD" column, change all instances of "scram-sha-256" (or whatever is in that column) to "trust". This will make it so PostgreSQL doesn't require a password.

In the Finder, click "Applications" then "PostgreSQL PGVERSION" and launch the "Reload Configuration" app. Click "OK" after you see "server signaled".

Next, to confirm the edit worked, launch the "pgAdmin" application from the same folder. Under "Browser", expand "Servers" and double click "PostgreSQL PGVERSION". When you are prompted for a password, leave it blank and click "OK". If you have successfully edited "pg_hba.conf", you can get in without a password.

On Linux, you should just install PostgreSQL using your favorite package manager, such as `yum`. (Consult the PostgreSQL section of *Prerequisites* in the main Installation guide for more info and command line examples). Find `pg_hba.conf` and set the authentication method to "trust" and restart PostgreSQL.

Install Solr

Solr 9.8.0 is required.

Follow the instructions in the "Installing Solr" section of *Prerequisites* in the main Installation guide.

Install Service Dependencies Using Docker Compose

To avoid having to install service dependencies like PostgreSQL or Solr directly on your localhost, there is the alternative of using the `docker-compose-dev.yml` file available in the repository root. For this option you need to have Docker and Docker Compose installed on your machine.

The `docker-compose-dev.yml` can be configured to only run the service dependencies necessary to support a Dataverse installation running directly on localhost. In addition to PostgreSQL and Solr, it also runs a SMTP server.

Before running the Docker Compose file, you need to update the value of the `DATAVERSE_DB_USER` environment variable to `postgres`. The variable can be found inside the `.env` file in the repository root. This step is required as the Dataverse installation script expects that database user.

To run the Docker Compose file, go to the Dataverse repository root, then run:

```
docker-compose -f docker-compose-dev.yml up -d --scale dev_dataverse=0
```

Note that this command omits the Dataverse container defined in the Docker Compose file, since Dataverse is going to be installed directly on localhost in the next section.

The command runs the containers in detached mode, but if you want to run them attached and thus view container logs in real time, remove the `-d` option from the command.

Data volumes of each dependency will be persisted inside the `docker-dev-volumes` folder, inside the repository root.

If you want to stop the containers, then run (for detached mode only, otherwise use `Ctrl + C`):

```
docker-compose -f docker-compose-dev.yml stop
```

If you want to remove the containers, then run:

```
docker-compose -f docker-compose-dev.yml down
```

If you want to run a single container (the mail server, for example) then run:

```
docker-compose -f docker-compose-dev.yml up dev_smtp
```

For a fresh installation, and before running the Software Installer Script, it is recommended to delete the docker-dev-env folder to avoid installation problems due to existing data in the containers.

8.34.4 Run the Dataverse Software Installer Script

Navigate to the directory where you cloned the Dataverse Software git repo change directories to the `scripts/installer` directory like this:

```
cd scripts/installer
```

Create a Python virtual environment, activate it, then install dependencies:

```
python3 -m venv venv
```

```
source venv/bin/activate
```

```
pip install psycopg2-binary
```

The installer will try to connect to the SMTP server you tell it to use. If you haven't used the Docker Compose option for setting up the dependencies, or you don't have a mail server handy, you can run `nc -l 25` in another terminal and choose "localhost" (the default) to get past this check.

Finally, run the installer (see also `README_python.txt` if necessary):

```
python3 install.py
```

Verify the Dataverse Software is Running

After the script has finished, you should be able to log into your Dataverse installation with the following credentials:

- http://localhost:8080
- username: dataverseAdmin
- password: admin

8.34.5 Configure Your Development Environment for Publishing

Run the following command:

```
curl http://localhost:8080/api/admin/settings/:DoiProvider -X PUT -d FAKE
```

This will disable DOI registration by using a fake (in-code) DOI provider. Please note that this feature is only available in Dataverse Software 4.10+ and that at present, the UI will give no indication that the DOIs thus minted are fake.

Developers may also wish to consider using *PermaLinks*

8.34.6 Configure Your Development Environment for GUI Edits

Out of the box, a JSF setting is configured for production use and prevents edits to the GUI (xhtml files) from being visible unless you do a full deployment.

It is recommended that you run the following command so that simply saving the xhtml file in Netbeans is enough for the change to show up.

```
asadmin create-system-properties "dataverse.jsf.refresh-period=1"
```

For more on JSF settings like this, see *Java Server Faces (JSF) Configuration Options*.

8.34.7 Next Steps

If you can log in to the Dataverse installation, great! If not, please see the *Troubleshooting* section. For further assistance, please see "Getting Help" in the *Introduction* section.

You're almost ready to start hacking on code. Now that the installer script has you up and running, you need to continue on to the *Tips* section to get set up to deploy code from your IDE or the command line.

8.35 Search Services

Dataverse supports configurable search services, allowing developers to integrate additional search engines dynamically. This guide outlines the design and provides details on how to use the interfaces and classes involved.

8.35.1 Design Overview

The configurable search services feature is designed to allow:

1. Dynamic addition of new search engines
2. Configuration of the Dataverse UI to use a specified search engine
3. Use of different search engines via the API
4. Discovery of installed search engines

8.35.2 Key Components

1. SearchService Interface

The `SearchService` interface is the core of the configurable search services. It defines the methods that any search engine implementation must provide. (The methods below are accurate as of this writing.)

```
public interface SearchService {
    String getServiceName();
    String getDisplayName();

    SolrQueryResponse search(DataverseRequest dataverseRequest, List<Dataverse>␣
↪dataverses, String query,
            List<String> filterQueries, String sortField, String sortOrder, int␣
↪paginationStart,
            boolean onlyDatatRelatedToMe, int numResultsPerPage, boolean␣
↪retrieveEntities, String geoPoint,
            String geoRadius, boolean addFacets, boolean addHighlights) throws␣
↪SearchException;

    default void setSolrSearchService(SearchService solrSearchService);
}
```

The interface allows you to provide a service name and display name, and to respond to the same search parameters that are normally sent to the Solr search engine.

The `setSolrSearchService` method is used by Dataverse to give your class a reference to the `SolrSearchService`, allowing your class to perform Solr queries as needed. (See the `ExternalSearchServices` for an example.)

2. ConfigurableSearchService Interface

The `ConfigurableSearchService` interface extends the `SearchService` interface and adds a method for Dataverse to set the `SettingsServiceBean`. This allows search services to be configurable through Dataverse settings.

```
public interface ConfigurableSearchService extends SearchService {
    void setSettingsService(SettingsServiceBean settingsService);
}
```

The `GetExternalSearchServiceBean` and `PostExternalSearchServiceBean` classes provide a use case for this.

3. JVM Options for Search Configuration

Dataverse uses two JVM options to configure the search functionality:

- `dataverse.search.services.directory`: Specifies the local directory where jar files with search engines (classes implementing the `SearchService` interface) can be found. Dataverse will dynamically load engines from this directory.
- `dataverse.search.default-service`: The `serviceName` of the service that should be used in the Dataverse UI.

Example configuration:

```
./asadmin create-jvm-options "-Ddataverse.search.services.directory=/var/lib/dataverse/
↪searchServices"
./asadmin create-jvm-options "-Ddataverse.search.default-service=solr"
```

Remember to restart your Payara server after modifying these JVM options for the changes to take effect.

4. Using Different Search Engines via API

The loaded search services can be discovered using the `/api/search/services` endpoint.

Queries can be made to different engines by including the optional `search_service=<serviceName>` query parameter.

Use of these endpoints is described for end users in the API Guide under *Search Services*.

8.35.3 Available Search Services

The class definitions for four example search services are included in the Dataverse repository. They are not included in the Dataverse .war file but can be built as three separate .jar files using

```
mvn clean package -DskipTests=true -Pexternal-search-get -Pexternal-search-post
```

or

```
mvn clean package -DskipTests=true -Ptrivial-search-examples
```

1. GetExternalSearchServiceBean

2. PostExternalSearchServiceBean

These classes implement the `ConfigurableSearchService` interface. They make a GET or POST call (respectively) to an external search engine that must return a JSON array of objects with "PID" (preferred) or "DOI" and "Distance" keys. The query sent to the external engine use the same query parameters as the Dataverse search API (GET) or have a JSON payload with those keys (POST). The results they return are then searched for using the solr search engine which enforces access control and provides the standard formatting expected by the Dataverse UI and API. The distance values are used to order the results, smallest distances first.

They can be configured via two settings each:

- GET
  - :GetExternalSearchUrl - the URL to send GET search queries to
  - :GetExternalSearchName - the display name to use for this configuration
- POST
  - :PostExternalSearchUrl - the URL to send POST search queries to
  - :PostExternalSearchName - the display name to use for this configuration

As these classes use PIDs as identifiers, they cannot reference collections or, unless file PIDs are enabled, files. Similar classes, or extensions of these classes could search by database ids instead, etc. to support the additional types.

3. GoldenOldiesSearchServiceBean

4. OddlyEnoughSearchServiceBean

These classes implement the `SearchService` interface. They are intended only as code examples and simple tests of the design and are not intended for production use. The former simply replaces the user query with a query for entities with a db id < 1000. It demonstrates how a class can leverage the solr engine and achieve results solely by modifying/replacing the user query. The latter only returns hits from the user's query that also have an odd database id. Since the filtering in the class changes the number of total hits available and pagination, this class demonstrates one way a developer can adjust those aspects of the Solr response.

8.35.4 Notes

1. Unless you use the Solr engine to provide access control, you must implement proper access control in your search engine
2. The design currently limits search results to be in the format returned by Solr and the hits are expected to be collections, datasets, or files - other classes are not supported.
3. Search services could be designed to completely replace Solr or to just support certain use cases (e.g. the external search classes only handling datasets).
4. While search services can be deployed as independent jar files, they currently import multiple Dataverse classes and, unlike exporters, cannot be built using just the Dataverse SPI.
5. As with other experimental features, we expect the `SearchService` interface may change over time as we learn about how people use it. Please keep in touch if you are developing search services.

8.36 Fixtures For Tests

Most Dataverse test fixtures are based on JSON files stored in the test resources of the codebase.

In addition, (as of Dataverse 6.11) you can use a generator utility to create dataset-centered fixtures programmatically. This is most useful for local integration and performance tests but may be of use for unit tests as well.

Contents:

- *Dataset Fixture Generator*
  - *Architecture*
    - *Fixture Builder*
    - *Recipes*
    - *Fixture Populator*
  - *Full Example*
  - *Basic Usage*
    - *Small Dataset*
    - *Skewed Variable Populations*
    - *Adding Variable Metadata*
  - *Persistence Usage*
  - *Discussion and Limitations*
    - *Benefits*
    - *Tradeoffs*
    - *Limitations*
    - *Unsupported Usage Scenarios*
  - *Extending The Fixture Generator*
    - *Add Recipes For Graph Shaping*
    - *Add Populator Behavior For Scalar Values*
    - *Change Builder For Wiring*
    - *Recommended Extension Path*
    - *Guidelines For Contributions*

8.36.1 Dataset Fixture Generator

The dataset fixture generator is a test utility for creating connected dataset entity graphs with configurable size and shape. It is located in the core testing utilities at `edu.harvard.iq.dataverse.util.testing.fixtures` and `edu.harvard.iq.dataverse.util.testing.recipes`.

The fixture generator is useful when tests need one or more datasets with many files, tabular files, variables, and optional variable metadata, while still keeping the test setup readable. It is primarily intended for integration and performance tests where hand-building entities would be too verbose, brittle, or too uniform to uncover ORM and serialization issues.

The generator creates an in-memory entity graph. Persisting that graph to a database is optional and requires the usual JPA persistence rules to be respected (see below).

Architecture

The fixture generator is built around three main concepts: a builder, recipes for it, and field populators. This separation keeps entity graph shape, relationship wiring, and scalar field population independent of each other.

Fixture Builder

The builder creates the connected *entity graph* by consuming recipes. It is responsible for:

- Creating the entities
- Wiring relationships
- Keeping both sides of relationships in sync where needed
- Returning a `DatasetFixture` with convenient references to generated objects

Recipes

Recipes *describe* the *shape* of the fixture's entity graph and should not manually wire entity relationships:

- How many files should exist?
- Which files are tabular?
- How many variables should a tabular file contain?
- Should variable metadata be created?

Available Recipes:

Recipes are composable using a fluent API and work together.

```
DatasetRecipe
  -> DatasetTypeRecipe
  -> VersionRecipe
       -> FileRecipe
            -> VariableSetRecipe
                 -> VariableMetadataRecipe
```

`DatasetRecipe`
Top-level recipe for creating a dataset fixture. It combines a `DatasetTypeRecipe` and a `VersionRecipe`.

`DatasetTypeRecipe`
Provides the dataset type assigned to the generated dataset. Can create a dataset type from scalar values or wrap an existing instance.

Note: the recipe provides the type object but does not persist it. Tests that persist generated fixtures must ensure the dataset type is managed before the dataset is flushed.

`VersionRecipe`
Describes the current dataset version. At the moment, this mainly means providing one or more file recipes.

`FileRecipe`
Describes file populations. A file recipe may create regular files or tabular files.

`VariableSetRecipe`
Describes how many variables to create for tabular files. It supports uniform and skewed variable populations.

`VariableMetadataRecipe`
Decides whether a `VariableMetadata` row should be created for a generated `(FileMetadata, DataVariable)` pair. At most one metadata row is generated for each such pair.

Fixture Populator

The populator fills scalar and non-relationship *fields*, which are not primarily about graph shape.

It sets values such as:

- Identifiers
- Timestamps
- File labels
- Content types
- Checksums
- Variable names
- Required fields
- Null-sensitive collections

The default *minimal* populator is conservative. It creates enough data for serialization and persistence tests, but it does not try to simulate fully realistic production metadata.

Full Example

The following example creates a small but non-uniform dataset fixture. It's suitable

- for a smoke test of a serializer,
- for an integration test with assertions on the result,
- for a performance test with benchmarking speed of different implementations, and other scenarios.

```
var recipe = DatasetRecipe.of(
    DatasetTypeRecipe.dataset(),
    VersionRecipe.of(
        FileRecipe.regular(20),
        FileRecipe.tabular(30,
            VariableSetRecipe
```

(continues on next page)

(continued from previous page)

```
                .byPredicate(VariableMetadataRecipe.byPredicate(ctx -> ctx.
↪variableIndex() < 5))
                .when(ctx -> ctx.fileIndex() % 10 == 0, 1_000)
                .otherwise(25)
    ))
);

DatasetFixture fixture = DatasetFixtureBuilder.builder()
    .recipe(recipe)
    .populator(FixturePopulator.minimal())
    .build();

JsonArrayBuilder files = JsonUtil.createArrayBuilder();

for (FileMetadata fileMetadata : fixture.fileMetadatas()) {
    files.add(JsonPrinter.json(fileMetadata.getDataFile(), fileMetadata, true));
}

var json = files.build();
```

This creates:

- 20 regular files
- 30 tabular files
  - some tabular files with 1,000 variables
  - other tabular files with 25 variables
  - variable metadata only for the first few variables in each tabular file

This helps exercise code paths that traverse files, file metadata, data tables, data variables, variable metadata. All of this happends without the need to pre-produce an enormous fixture as a JSON file. Its deterministic nature allows running the test anywhere without depending on seeded randomness, offering reliable and reproducible results.

Basic Usage

Small Dataset

This example creates:

- one dataset
- one current version
- 10 tabular files
- 10 variables per tabular file
- 1 regular file

```
var recipe = DatasetRecipe.of(
    DatasetTypeRecipe.dataset(),
    VersionRecipe.of(
        FileRecipe.tabular(10, VariableSetRecipe.uniform(10)),
        FileRecipe.regular(1)
```

(continues on next page)

(continued from previous page)

```
    )
);

DatasetFixture fixture = DatasetFixtureBuilder.builder()
    .recipe(recipe)
    .populator(FixturePopulator.minimal())
    .build();

Dataset dataset = fixture.dataset();
DatasetVersion version = fixture.currentVersion();
```

Skewed Variable Populations

Skewed data is useful for performance testing because real datasets are rarely uniform. Some files may have only a few variables, while others may be very large.

This example creates 500 tabular files:

- one dataset
- one current version
- 500 tabular files
  - every 100th file receives 100,000 variables
  - every 10th file receives 10,000 variables
  - all others receive 250 variables

```
var variables = VariableSetRecipe.byPredicate()
    .when(ctx -> ctx.fileIndex() % 100 == 0, 100_000)
    .when(ctx -> ctx.fileIndex() % 10 == 0, 10_000)
    .otherwise(250);

var recipe = DatasetRecipe.of(
    DatasetTypeRecipe.dataset(),
    VersionRecipe.of(
        FileRecipe.tabular(500, variables)
    )
);

DatasetFixture fixture = DatasetFixtureBuilder.builder()
    .recipe(recipe)
    .build();
```

Adding Variable Metadata

Variable Metadata is optional and controlled by `VariableMetadataRecipe`. The metadata recipe is evaluated for each generated `(FileMetadata, DataVariable)` pair. This matters because `VariableMetadata` is versioned indirectly through `FileMetadata`.

No variable metadata (default):

```
VariableSetRecipe.uniform(1_000)
- or -
VariableSetRecipe.uniform(1_000, VariableMetadataRecipe.noop())
```

Metadata for every variable:

```
VariableSetRecipe.uniform(1_000, VariableMetadataRecipe.always())
```

Metadata for selected variables:

```
VariableSetRecipe.uniform(1_000, VariableMetadataRecipe.byPredicate(ctx -> ctx.
↪variableIndex() % 10 == 0))
```

Persistence Usage

The generator creates an in-memory entity graph. Persisting that graph is optional and follows normal JPA rules.

When persisting a generated fixture to a database, remember that not all relationships cascade from `Dataset` to every object. In particular, `DataFile` instances usually need to be persisted explicitly before persisting the dataset graph. The `DatasetType` must also be managed, either by persisting the generated type or by looking up an existing one in the same persistence context.

A typical persistence sequence is:

```
jpa.inTransactionVoid(em -> {
    em.persist(fixture.datasetType());
    for (DataFile dataFile : fixture.dataFiles()) {
        em.persist(dataFile);
    }
    em.persist(fixture.dataset());
});
```

The exact order may evolve as the fixture generator grows, may depend on the exact usage scenario, and is influenced by the evolution of the entity classes themselves, but the important point is:
Shared/reference entities and non-cascaded entities must be managed (persisted) before the dataset graph is flushed.

Discussion and Limitations

Benefits

1. **Readable scenarios:** tests describe intent at a high level. For example: `FileRecipe.tabular(500, VariableSetRecipe.uniform(1_000))` is easier to understand than manually creating thousands of entities.
2. **Composable graph shape:** different recipes can be combined to describe mixed datasets.
3. **Deterministic output:** the build context carries fixture-wide values such as sequence and timestamp, making generated data easier to debug and compare.
4. **Reduced boilerplate:** relationship wiring and null-sensitive defaults are centralized.
5. **Better performance testing:** skewed fixtures can expose ORM issues that uniform data may hide, such as N+1 query expansion over large variable collections.
6. **Serialization safety:** the minimal populator initializes fields and collections that serializers commonly traverse.

Tradeoffs

1. **More concepts to learn:** developers need to understand builders, recipes, populators, and resulting fixture objects vs. a static factory.
2. **Not a full production object factory:** the minimal populator creates safe test data, not necessarily realistic production data.
3. **Persistence still requires care:** some entities must be persisted explicitly because the production model does not cascade every relationship.
4. **Hardcoded defaults:** the minimal populator uses deterministic placeholder values, tests that need realistic metadata should provide a custom populator.

Limitations

1. **Minimalistic:** The current fixture generator is intentionally minimal.
2. **Single dataset version only:** the fixture currently models one current dataset version and does not generate multiple versions.
3. **No version evolution recipes:** there is no support yet for deriving later versions from earlier versions, modeling change over time.
4. **Limited dataset metadata:** dataset fields and metadata blocks are not generated in detail.
5. **Simple dataset type handling:** a `DatasetType` can be generated or supplied, but persistence of shared types is still the responsibility of the test.
6. **No persistence manager:** the fixture system builds graphs, but it does not yet provide a dedicated persister that knows the correct persistence order.
7. **One table per tabular file:** tabular files currently get one `DataTable`. The domain model can allow more, but the fixture generator does not expose that yet.
8. **One variable group per tabular file:** each non-empty tabular file currently gets one `VarGroup` containing all variables, there is no `VarGroupRecipe` yet.
9. **Limited variable metadata content:** variable metadata can be present or absent, but the minimal populator only fills basic scalar values.

10. **No category or statistics recipes:** the fixture generator does not yet provide recipes for variable categories, summary statistics, invalid ranges, or category metadata.

Unsupported Usage Scenarios

The following scenarios are not yet directly expressible:

- multiple dataset versions sharing the same `DataFile` objects
- metadata-only changes between versions
- version-specific `VariableMetadata` changes across versions
- files added or removed between versions
- multiple `DataTable` objects per file
- different variable group distributions per file
- weighted random or seeded random file populations
- Zipf-like or heavy-tail distributions as first-class recipes
- realistic dataset field metadata
- fixture graphs that mimic a fully published dataset lifecycle

Extending The Fixture Generator

When extending the fixture generator, first decide which responsibility your change belongs to.

Add Recipes For Graph Shaping

Use a new recipe when the test needs to describe what shape should be created.

Examples:

- number of var groups
- number of data tables per file
- whether categories should exist
- how many variables receive summary statistics
- how versions evolve over time

Recipe changes usually belong in the `edu.harvard.iq.dataverse.util.testing.recipes` package.

Add Populator Behavior For Scalar Values

Use a new or custom populator when entities should be filled differently, but the graph shape is the same. Extend the populator interface if new types of scalar data are required.

Examples:

- more realistic file names
- different content types
- richer variable labels

- custom checksums
- realistic variable metadata text

Populator changes usually belong in the `edu.harvard.iq.dataverse.util.testing.fixtures` package.

Change Builder For Wiring

Change the builder when new relationships must be created or maintained.

Examples:

- adding support for `VariableCategory`
- wiring category metadata
- creating multiple data tables per file
- linking version-evolved file metadata back to shared data files

Builder changes should be kept small and split into helper methods where possible.

Recommended Extension Path

A practical roadmap for further evolution is:

1. Add a `VarGroupRecipe` to control group count and membership.
2. Add category and summary statistic recipes for variable-level enrichment.
3. Add a fixture persister that knows the correct persistence order.
4. Add version evolution recipes for multi-version datasets.
5. Add richer dataset metadata generation.
6. Add (seeded!) random distribution recipes if a deterministic skew is not enough.
7. Add fuzzy testing by generating fixtures with targeted chaos.

Guidelines For Contributions

1. Keep recipes declarative: recipes should describe shape, not manually wire entity relationships.
2. Keep populators focused: populators should fill fields, not decide how many entities exist.
3. Keep builders responsible for wiring: relationship consistency belongs in the builder.
4. Prefer deterministic generation: deterministic data makes performance tests easier to reproduce and debug.
5. Avoid hiding persistence requirements: if an entity must be persisted before another, document it clearly or add a dedicated persister.
6. Start minimal: add the smallest recipe or populator extension needed for the scenario. Avoid making the DSL generic before there is a concrete test need.

8.37 Performance Testing

8.37.1 Introduction

Performance tests measure how your application behaves under load, focusing on execution time, resource consumption, and database efficiency. Unlike *unit tests*, which verify isolated logic, or *integration* or *API tests*, which validate component interactions and full request lifecycles, performance tests quantify *how fast* operations complete and *how many* database queries they trigger.

8.37.2 Running Performance Tests

Performance tests are excluded from the default test run to save CI/CD time and local resources. To execute them, use the Maven `verify` lifecycle phase and override the `it.groups` property:

```
mvn verify -Dit.groups=performance
```

Note

The `it.groups` property accepts a comma-separated list. You can combine groups (e.g., `-Dit.groups=integration,performance`) as necessary. However, it is highly recommended to run them in isolation due to their computational intensity and sensitivity to system load.

8.37.3 Testing database-bound code

Performance tests for code relying on retrieving entities from a database are essential for catching regressions in ORM efficiency. They can identify N+1 query problems or ensure that heavy data processing pipelines (e.g., exporting large datasets) remain responsive as the codebase evolves.

Prerequisites

Any tests around database-bound code rely on Testcontainers to spin up ephemeral database instances. Avoiding in-memory databases for such tests allow for more realistic testing as seen in actual deployments. Consequently, you must have **Docker** installed and running, allowing Testcontainer to start a PostgreSQL server.

- If you use a local Docker daemon, ensure it has sufficient memory allocated (typically 1GB+ is recommended for running Postgres containers alongside your tests).
- If your Docker daemon runs remotely, ensure the `DOCKER_HOST` environment variable is correctly configured in your shell so Testcontainers can locate it.

The automated testing setup will look up a system property `postgresql.server.version` to determine which container image tag to use. The property is injected from `pom.xml` by Maven Failsafe and use a reasonable fallback value if missing. To test with a different version of PostgreSQL, you may set the Maven property `postgresql.server.version` for a run.

Example

Performance test classes must follow specific conventions to be discovered and executed correctly:

1. **Package Location:**
   Place your test class in `src/test/java`, mirroring the package structure of the code you want to test (e.g., `edu.harvard.iq.dataverse.export`). This placement grants the test class access to package private members in `src/main/java`, which is often necessary when testing internal services directly without going through the full API layer.
2. **Naming Convention:**
   Name the class `*IT.java` so that the Maven Failsafe plugin automatically picks it up during the `integration-test` phase.
3. **Setup Annotation:**
   Annotate the class with `@JpaPerformanceTest` to have everything set up automatically for you. A `JpaEntityManagerService` will be injected into a static class field for you, allowing interaction with a JPA Entity Manager.

Below is a minimal, generic example `SamplePerformanceIT` demonstrating the structure and how to run a transaction with or without a return value.

```
// Single annotation for automatic setup of
//     1) basic tags for JUnit groups,
//     2) shared PostgreSQL server via Testcontainers, and
//     3) creation and injection of JPA entity manager service.
@JpaPerformanceTest
class SamplePerformanceIT {

    static JpaEntityManagerService jpa;

    @BeforeAll
    static void setUp() {
        // A manual start is necessary to allow you to selectively enable service␣
↪features as necessary
        jpa.start();

        // inTransactionVoid: Use this when you only need to execute database operations
        // (e.g., persisting test fixtures) without returning a value.
        jpa.inTransactionVoid(em -> {
            // EntityManager em is provided here.
            // em.persist(myEntity);
        });
    }

    @Test
    void shouldMeasureOperationPerformance() {
        // Clear any previous query statistics
        QueryCountHolder.clear();
        Instant start = Instant.now();

        // inTransaction: Use this when your operation returns a result that needs
        // to be asserted or measured.
        Object result = jpa.inTransaction(em -> {
            // Execute your performance-critical operation using the EntityManager.
```

(continues on next page)

(continued from previous page)

```
            // return result;
            return null; // Placeholder
        });

        Instant end = Instant.now();
        assertNotNull(result);

        // Retrieve and log ORM statistics
        QueryCount count = QueryCountHolder.getGrandTotal();
        System.out.println("Elapsed ms: " + start.until(end, ChronoUnit.MILLIS));
        System.out.println("Total queries: " + count.getTotal());
        System.out.println("Select queries: " + count.getSelect());
        System.out.println("Insert queries: " + count.getInsert());
        System.out.println("Update queries: " + count.getUpdate());
        System.out.println("Delete queries: " + count.getDelete());
    }
}
```

Understanding JpaEntityManagerService

The `JpaEntityManagerService` class abstracts away the boilerplate required to set up a JPA environment for testing. Here is what it does under the hood:

1. **Automatic PostgreSQL Server Setup:**
   The involved JUnit Test Extension makes sure to create a single server instance to speed up test setups. Nonetheless, any test class will run within its own database on the server, guaranteeing test database isolation.

2. **Automatic Schema Generation:**
   When you call `.start()` on a `JpaEntityManagerService` instance, it initializes an EclipseLink `EntityManagerFactory` configured to automatically generate the database schema (`schema-generation.database.action=create`). This guarantees that every test run begins with a pristine database structure derived directly from your current JPA entity mappings. You do not need to run Flyway migrations or seed the database beforehand.

3. **Transaction Management:**
   The service handles the lifecycle of JPA transactions automatically. You simply pass a lambda to `inTransaction()` or `inTransactionVoid()`. The service will:

   1. Create an `EntityManager` and begin a transaction.
   2. Execute your lambda.
   3. Commit the transaction on success, or roll it back if a `RuntimeException` is thrown.
   4. Close the `EntityManager` in a `finally` block to prevent resource leaks.

4. **Query Statistics via Wrapped DataSource:**
   To make it easy to profile ORM behavior, `JpaEntityManagerService` wraps the underlying PostgreSQL `DataSource` using a proxy that intercepts all SQL statements.

   By default, the proxy tracks query counts, which you can retrieve via `QueryCountHolder.getGrandTotal()`. This provides immediate, programmatic insight into database efficiency without needing to parse verbose SQL logs. It is particularly useful for:

   - Verifying that a batch operation executes in a single query rather than a loop.
   - Catching N+1 query problems by asserting on the number of `SELECT` statements.

Advanced Usage: The default service only tracks query counts. If you need detailed SQL logging (including bound parameters) or custom execution metrics, you can extend `JpaEntityManagerService` and register additional `StatementListener` implementations on the `ProxyDataSourceBuilder` during initialization.

CHAPTER

NINE

CONTAINER GUIDE

Contents:

9.1 Introduction

Dataverse in containers!

Contents:

- *Intended Audience*
- *Getting Help*
- *Helping with the Containerization Effort*

9.1.1 Intended Audience

This guide is intended for anyone who wants to run Dataverse in containers. This is potentially a wide audience, from sysadmins interested in running Dataverse in production in containers to contributors working on a bug fix. See *Running Dataverse in Docker* for various scenarios and please let us know if your use case is not covered.

9.1.2 Getting Help

Please ask in #containers at https://chat.dataverse.org

Alternatively, you can try one or more of the channels under *Getting Help*.

9.1.3 Helping with the Containerization Effort

In 2023 the Containerization Working Group started meeting regularly. All are welcome to join! We talk in #containers at https://chat.dataverse.org and have a regular video call. For details, please visit https://ct.gdcc.io

9.2 Running Dataverse in Docker

Contents:

9.2.1 Production

Warning

The *Demo or Evaluation* tutorial is **NOT SECURE BY DEFAULT**. It uses public, hardcoded passwords and secrets for demonstration purposes only.

If you use the demo as a structural template, you MUST replace all default secrets before deploying your instance. Failure to do so will result in a vulnerable production environment.

Contents:

- *Status*
- *Limitations*
- *How to Help*
- *Alternatives*

Status

As of Dataverse 6.8, when we introduced image tagging per version (see the *Supported Image Tags* section for the *application image*), we feel that the images described in this guide are ready for production use. Enjoy!

The images and the documentation is not perfect, of course.

For now, we recommend following the *Demo or Evaluation* as a structural template. Note that instead of "latest" you might want to select a specific version. Again see *Supported Image Tags*.

The Dataverse guides were originally written with a non-Docker installation in mind so we'd like rewrite them with both Docker and non-Docker in mind. This is a big job, obviously. We know we'd like to write more about ports. We'd like to explain how to set up Rserve. Etc., etc.

To talk about your ideas for making the images and docs better for production, please feel free to join the containers for production topic or join a working group meeting (see *Helping with the Containerization Effort*).

Limitations

- Multiple apps servers are not supported. See *Multiple App Servers* for more on this topic.

How to Help

Please try the images (see *Demo or Evaluation*) and give feedback (see *Helping with the Containerization Effort*)!

Alternatives

The traditional (non-Docker) installation method is described in the *Installation Guide*.

9.2.2 Demo or Evaluation

In the following tutorial we'll walk through spinning up Dataverse in containers for demo or evaluation purposes.

Contents:

- *Quickstart*
- *Stopping and Starting the Containers*
- *Deleting Data and Starting Over*
- *Setting Up for a Demo*
  - *Starting Fresh*
  - *Creating and Running a Demo Persona*
- *Smoke Testing*
- *Further Configuration*
  - *JVM Options/MicroProfile Config*
  - *Database Settings*
  - *Root Collection Customization (Alias, Name, etc.)*
  - *Multiple Languages*
  - *PID Providers*
  - *File Previewers*
  - *Additional Metadata Blocks*
- *Next Steps*
- *About the Containers*
  - *Container List*
  - *Tags and Versions*
- *Troubleshooting*
  - *Hardware and Software Requirements*

    - *Bootstrapping Did Not Complete*
- *Wrapping Up*
    - *Deleting the Containers and Data*
- *Giving Feedback*
- *Getting Help*

Quickstart

First, let's confirm that we can get Dataverse running on your system.

- Download `compose.yml`
- Run `docker compose up` in the directory where you put `compose.yml`
- Visit http://localhost:8080 and try logging in:
    - username: dataverseAdmin
    - password: admin1

If you can log in, great! Please continue through the tutorial. If you have any trouble, please consult the sections below on troubleshooting and getting help.

Stopping and Starting the Containers

Let's practice stopping the containers and starting them up again. Your data, stored in a directory called `data`, will remain intact

To stop the containers hit `Ctrl-c` (hold down the `Ctrl` key and then hit the `c` key).

To start the containers, run `docker compose up`.

Deleting Data and Starting Over

Again, data related to your Dataverse installation such as the database is stored in a directory called `data` that gets created in the directory where you ran `docker compose` commands.

You may reach a point during your demo or evaluation that you'd like to start over with a fresh database. Simply make sure the containers are not running and then remove the `data` directory. Now, as before, you can run `docker compose up` to spin up the containers.

Setting Up for a Demo

Now that you are familiar with the basics of running Dataverse in containers, let's move on to a better setup for a demo or evaluation.

Starting Fresh

For this exercise, please start fresh by stopping all containers and removing the `data` directory.

Creating and Running a Demo Persona

Previously we used the "dev" persona to bootstrap Dataverse, but for security reasons, we should create a persona more suited to demos and evaluations.

Edit the `compose.yml` file and look for the following section.

```
bootstrap:
  container_name: "bootstrap"
  image: gdcc/configbaker:latest
  restart: "no"
  environment:
    - TIMEOUT=3m
  command:
    - bootstrap.sh
    - dev
    #- demo
  #volumes:
  #  - ./demo:/scripts/bootstrap/demo
  networks:
    - dataverse
```

Comment out "dev" and uncomment "demo".

Uncomment the "volumes" section.

Create a directory called "demo" and copy `init.sh` into it. You are welcome to edit this demo init script, customizing the final message, for example.

Note that the init script contains a key for using the admin API once it is blocked. You should change it in the script from "unblockme" to something only you know.

Now run `docker compose up`. The "bootstrap" container should exit with the message from the init script and Dataverse should be running on http://localhost:8080 as before during the quickstart exercise.

One of the main differences between the "dev" persona and our new "demo" persona is that we are now running the setup-all script without the `--insecure` flag. This makes our installation more secure, though it does block "admin" APIs that are useful for configuration.

Smoke Testing

At this point, please try the following basic operations within your installation:

- logging in as dataverseAdmin (password "admin1")
- publishing the "root" collection (dataverse)
- creating a collection
- creating a dataset
- uploading a data file
- publishing the dataset

If anything isn't working, please see the sections below on troubleshooting, giving feedback, and getting help.

Further Configuration

Now that we've verified through a smoke test that basic operations are working, let's configure our installation of Dataverse.

Please refer to the *Configuration* section of the Installation Guide for various configuration options.

Below we'll explain some specifics for configuration in containers.

JVM Options/MicroProfile Config

JVM Options can be configured under `JVM_ARGS` in the `compose.yml` file. Here's an example:

```
environment:
  JVM_ARGS: -Ddataverse.files.storage-driver-id=file1
```

Some JVM options can be configured as environment variables. For example, you can configure the database host like this:

```
environment:
  DATAVERSE_DB_HOST: postgres
```

We are in the process of making more JVM options configurable as environment variables. Look for the term "MicroProfile Config" in under *Configuration* in the Installation Guide to know if you can use them this way.

There is a final way to configure JVM options that we plan to deprecate once all JVM options have been converted to MicroProfile Config. Look for "magic trick" under "tunables" at *Application Image* for more information.

Database Settings

Generally, you should be able to look at the list of *Database Settings* and configure them but the "demo" persona above secured your installation to the point that you'll need an "unblock key" to access the "admin" API and change database settings.

In the example below of configuring *:FooterCopyright* we use the default unblock key of "unblockme" but you should use the key you set above.

```
curl -X PUT -d ", My Org" "http://localhost:8080/api/admin/settings/:FooterCopyright?
unblock-key=unblockme"
```

One you make this change it should be visible in the copyright in the bottom left of every page.

Root Collection Customization (Alias, Name, etc.)

Before running `docker compose up` for the first time, you can customize the root collection by placing a JSON file in the right place.

First, in the "demo" directory you created (see *Creating and Running a Demo Persona*), create a subdirectory called "config":

```
mkdir demo/config
```

Next, download `dataverse-complete.json` and put it in the "config" directory you just created. The contents of your "demo" directory should look something like this:

```
% find demo
demo
demo/config
demo/config/dataverse-complete.json
demo/init.sh
```

Edit `dataverse-complete.json` to have the values you want. You'll want to refer to *Update a Dataverse Collection* in the API Guide to understand the format. In that documentation you can find optional parameters as well.

To test your JSON file, run `docker compose up`. Again, this only works when you are running `docker compose up` for the first time. (You can always start over. See *Deleting Data and Starting Over*.)

Multiple Languages

Generally speaking, you'll want to follow *Internationalization* in the Installation Guide to set up multiple languages. (You need to create your own "languages.zip" file, for example.) Here will give you guidance specific to this demo tutorial. We'll be setting up a toggle between English and French.

First, edit the `compose.yml` file and uncomment the following line:

```
#-Ddataverse.lang.directory=/dv/lang
```

Next, upload "languages.zip" to the "loadpropertyfiles" API endpoint as shown below. This will place files ending in ".properties" into the `/dv/lang` directory configured above.

Please note that we are using a slight variation on the command in the instructions above, adding the unblock key we created above:

```
curl "http://localhost:8080/api/admin/datasetfield/loadpropertyfiles?
unblock-key=unblockme" -X POST --upload-file /tmp/languages/languages.zip -H
"Content-Type:  application/zip"
```

Next, set up the UI toggle between English and French, again using the unblock key:

```
curl "http://localhost:8080/api/admin/settings/:Languages?unblock-key=unblockme" -X PUT
-d '[{"locale":"en","title":"English"},{"locale":"fr","title":"Français"}]'
```

Stop and start the Dataverse container in order for the language toggle to work.

PID Providers

Dataverse supports multiple Persistent ID (PID) providers. The `compose.yml` file uses the Permalink PID provider. Follow *Persistent Identifiers and Publishing Datasets* to reconfigure as needed.

File Previewers

By default, all available file previewers are enabled (see *File Previews* in the User Guide for details). Specifically, we enable all the previewers that are available in the trivadis/dataverse-previewers-provider image (code). You can run the following command to see a list of available previewers:

```
docker run --rm trivadis/dataverse-deploy-previewers:latest previewers
```

You should expect to see output like this:

```
name     description
--------------------------
text     Read the text file.
html     View the html file.
...
```

If you want to specify fewer previewers, you can edit the `compose.yml` file. Uncomment "INCLUDE_PREVIEWERS" and list the previewers you want, separated by commas, like this:

```
INCLUDE_PREVIEWERS=text,html,pdf,csv
```

Additional Metadata Blocks

Metadata fields such as "Title" are part of a metadata block such as "Citation". See *Metadata References* in the User Guide for the metadata blocks that ship with Dataverse.

At a high level, we will be loading a metadata block and then adjusting our Solr config to know about it.

Care should be taken when adding additional metadata blocks. There is no way to preview or delete a metadata block so please use a throwaway environment.

Metadata References lists some experimental metadata blocks. In the example below, we'll use the CodeMeta block.

First, download a metadata block or create one by following *Metadata Customization* in the Admin Guide.

Load the metadata block like this:

```
curl http://localhost:8080/api/admin/datasetfield/load -H "Content-type:  text/
tab-separated-values" -X POST --upload-file codemeta.tsv
```

Next, reconfigure Solr to know about the new metadata block.

You can back up your existing Solr schema like this:

```
cp docker-dev-volumes/solr/data/data/collection1/conf/schema.xml docker-dev-volumes/solr/
data/data/collection1/conf/schema.xml.orig
```

You can see the existing fields Solr knows about like this:

```
curl http://localhost:8983/solr/collection1/schema/fields
```

Update your Solr schema with the following command:

```
curl http://localhost:8080/api/admin/index/solr/schema | docker run -i --rm -v ./
docker-dev-volumes/solr/data:/var/solr gdcc/configbaker:unstable update-fields.sh /var/
solr/data/collection1/conf/schema.xml
```

Then, reload Solr:

```
curl "http://localhost:8983/solr/admin/cores?action=RELOAD&core=collection1"
```

You can get a diff of your old and new Solr schema like this:

```
diff docker-dev-volumes/solr/data/data/collection1/conf/schema.xml.orig
docker-dev-volumes/solr/data/data/collection1/conf/schema.xml
```

You should be able to see the new fields from the metadata block you added in the following output:

```
curl http://localhost:8983/solr/collection1/schema/fields
```

At this point you can proceed with testing the metadata block in the Dataverse UI. First you'll need to enable it for a collection (see *General Information* in the User Guide section about collections). Afterwards, create a new dataset, save it, and then edit the metadata for that dataset. Your metadata block should appear.

Next Steps

From here, you are encouraged to continue poking around, configuring, and testing. You probably spend a lot of time reading the *Configuration* section of the Installation Guide.

Please consider giving feedback using the methods described below. Good luck with your demo!

About the Containers

Now that you've gone through the tutorial, you might be interested in the various containers you've spun up and what they do.

Container List

If you run `docker ps`, you'll see that multiple containers are spun up in a demo or evaluation. Here are the most important ones:

- dataverse
- postgres
- solr
- smtp
- bootstrap

Most are self-explanatory, and correspond to components listed under *Prerequisites* in the (traditional) Installation Guide, but "bootstrap" refers to *Config Baker Image*.

Additional containers are used in development (see *Development Usage*), but for the purposes of a demo or evaluation, fewer moving (sometimes pointy) parts are included.

Tags and Versions

The compose file references a tag called "latest", which corresponds to the latest released version of Dataverse. This means that if a release of Dataverse comes out while you are demo'ing or evaluating, the version of Dataverse you are using could change if you do a `docker pull`. Feel free to change it to a specific version to avoid this. For more on available tags, see supported tags section for *Application* and *Config Baker* images.

Once Dataverse is running, you can check which version you have through the normal methods:

- Check the bottom right in a web browser.
- Check http://localhost:8080/api/info/version via API.

Troubleshooting

Hardware and Software Requirements

- 8 GB RAM (if not much else is running)
- Mac, Linux, or Windows (experimental)
- Docker

Windows support is experimental but we are very interested in supporting Windows better. Please report bugs (see *Helping with the Containerization Effort*).

Bootstrapping Did Not Complete

In the compose file, try increasing the timeout for the bootstrap container:

```
environment:
  - TIMEOUT=10m
```

As described above, you'll want to stop containers, delete data, and start over with `docker compose up`. To make sure the increased timeout is in effect, you can run `docker logs bootstrap` and look for the new value in the output:

```
Waiting for http://dataverse:8080 to become ready in max 10m.
```

Wrapping Up

Deleting the Containers and Data

If you no longer need the containers because your demo or evaluation is finished and you want to reclaim disk space, run `docker compose down` in the directory where you put `compose.yml`.

You might also want to delete the `data` directory, as described above.

Giving Feedback

Your feedback is extremely valuable to us! To let us know what you think, please see *Helping with the Containerization Effort*.

Getting Help

Please do not be shy about reaching out for help. We very much want you to have a pleasant demo or evaluation experience. For ways to contact us, please see *Getting Help*.

9.2.3 Editing Metadata Blocks

Contents:

- *Intro*
- *Status*

Intro

The Admin Guide has a section on *Metadata Customization* and suggests running Dataverse in containers (Docker) for this purpose.

Status

For now, please see *Demo or Evaluation*, which should also provide a suitable Dockerized Dataverse environment.

9.2.4 GitHub Action

Contents:

- *Intro*
- *Use Cases*

Intro

A GitHub Action is under development that will spin up a Dataverse instance within the context of GitHub CI workflows: https://github.com/gdcc/dataverse-action

Use Cases

Use cases for the GitHub Action include:

- Testing *Client Libraries* that interact with Dataverse APIs
- Testing *Integrations* of third party software with Dataverse

9.2.5 Frontend Development

Contents:

- *Intro*

Intro

The frontend (web interface) of Dataverse is being decoupled from the backend. This evolving codebase has its own repo at https://github.com/IQSS/dataverse-frontend which includes docs and scripts for running the backend of Dataverse in Docker.

9.2.6 Backend Development

Contents:

- *Intro*

Intro

See *Development Usage*.

9.3 Development Usage

Please note! This Docker setup is not for *production*!

Contents:

- *Quickstart*
- *Intro*
- *Building*
- *Running*
- *Viewing Logs*
  - *Accessing Harvesting Log Files*
- *Redeploying*
  - *IDE Triggered Code Re-Deployments*
  - *IDE Triggered Non-Code Re-Deployments*
  - *Fast Redeploy (Command-Line)*

- *Exploring the Database*
- *Using a Debugger*
- *Building Your Own Base Image*

9.3.1 Quickstart

See *Quickstart*.

9.3.2 Intro

Assuming you have Docker, Docker Desktop, Moby or some remote Docker host configured, up and running from here on. Also assuming you have Java and Maven installed, as you are at least about to develop code changes.

To test drive these local changes to the Dataverse codebase in a containerized application server (and avoid the setup described in *Development Environment*), you must a) build the application and b) run it in addition to the necessary dependencies. (Which might involve building a new local version of the *Config Baker Image*.)

9.3.3 Building

To build the *application* and *config baker image*, run the following command:

```
mvn -Pct clean package
```

Once this is done, you will see images `gdcc/dataverse:unstable` and `gdcc/configbaker:unstable` available in your Docker cache.

Note: This will skip any unit tests. If you have built the code before for testing, etc. you might omit the `clean` to avoid recompiling.

Note: Also we have a `docker-compose-dev.yml` file, it's currently not possible to build the images without invoking Maven. This might change in the future.

9.3.4 Running

After building the app and config baker image containing your local changes to the Dataverse application, you want to run it together with all dependencies. There are four ways to do this (commands executed at root of project directory):

Table 1: Cheatsheet: Running Containers

	Using Maven	Using Compose
In foreground	`mvn -Pct docker:run`	`docker compose -f docker-compose-dev.yml up`
In background	`mvn -Pct docker:start`	`docker compose -f docker-compose-dev.yml up -d`

Both ways have their pros and cons:

Table 2: Decision Helper: Fore- or Background?

	Pros	Cons
Foreground		
	Logs scroll by when interacting with API / UI To stop all containers simply hit `Ctrl+C`	Lots and lots of logs scrolling by Must stop all containers to restart
Background		
	No logs scrolling by Easy to replace single containers	No logs scrolling by Stopping containers needs an extra command

In case you want to concatenate building and running, here's a cheatsheet for you:

Table 3: Cheatsheet: Building and Running Containers

	Using Maven	Using Compose
In foreground	`mvn -Pct package docker:run`	`mvn -Pct package && docker compose -f docker-compose-dev.yml up`
In background	`mvn -Pct package docker:start`	`mvn -Pct package && docker compose -f docker-compose-dev.yml up -d`

Once all containers have been started, you can check if the application was deployed correctly by checking the version at http://localhost:8080/api/info/version or watch the logs.

Note: To stop all containers you started in background, invoke `mvn -Pct docker:stop` or `docker compose -f docker-compose-dev.yml down`.

Check that you can log in to http://localhost:8080 using user `dataverseAdmin` and password `admin1`.

You can also access the Payara Admin Console if needed, which is available at http://localhost:4848. To log in, use user `admin` and password `admin`. As a reminder, the application container is for development use only, so we are exposing the admin console for testing purposes. In a production environment, it may be more convenient to leave this console unopened.

Note that data is persisted in `./docker-dev-volumes` in the root of the Git repo. For a clean start, you should remove this directory before running the `mvn` commands above.

9.3.5 Viewing Logs

In case you started containers in background mode (see *Running*), you can use the following commands to view and/or watch logs from the containers.

The safe bet for any running container's logs is to lookup the container name via `docker ps` and use it in `docker logs <name>`. You can tail logs by adding `-n` and follow them by adding `-f` (just like `tail` cmd). See `docker logs --help` for more.

Alternatives:

- In case you used Maven for running, you may use `mvn -Pct docker:logs -Ddocker.filter=<service name>`.

- If you used Docker Compose for running, you may use `docker compose -f docker-compose-dev.yml logs <service name>`. Options are the same.

Accessing Harvesting Log Files

1. Open a terminal and access the Dataverse container.

Run the following command to access the Dataverse container:

```
docker exec -it dev_dataverse bash
```

This command opens an interactive shell within the dev_dataverse container.

2. Navigate to the log files directory.

Once inside the container, navigate to the directory where Dataverse logs are stored:

```
cd /opt/payara/appserver/glassfish/domains/domain1/logs
```

This directory contains various log files, including those relevant to harvesting.

3. Create a directory for copying files.

Create a directory where you'll copy the files you want to access on your local machine:

```
mkdir /dv/filesToCopy
```

This will create a new folder named filesToCopy inside /dv.

4. Copy the files to the new directory.

Copy all files from the current directory to the newly created filesToCopy directory:

```
cp * /dv/filesToCopy
```

This command copies all files in the logs directory to /dv/filesToCopy.

5. Access the files on your local machine.

On your local machine, the copied files should appear in the following directory:

```
docker-dev-volumes/app/data/filesToCopy
```

9.3.6 Redeploying

The safest and most reliable way to redeploy code is to stop the running containers (with Ctrl-c if you started them in the foreground) and then build and run them again with `mvn -Pct clean package docker:run`. Safe, but also slowing down the development cycle a lot.

Triggering redeployment of changes using an IDE can greatly improve your feedback loop when changing code. You have at least two options:

1. Use builtin features of IDEs or IDE plugins from Payara.
2. Use a paid product like JRebel.

The main differences between the first and the second options are support for hot deploys of non-class files and limitations in what the JVM HotswapAgent can do for you. Find more details in a blog article by JRebel.

IDE Triggered Code Re-Deployments

To make use of builtin features or Payara IDE Tools (option 1), please follow steps below. Note that using this method, you may redeploy a complete WAR or single methods. Redeploying WARs supports swapping and adding classes and non-code materials, but is slower (still faster than rebuilding containers). Hotswapping methods requires using JDWP (Debug Mode), but does not allow switching non-code material or adding classes.

1. Download the version of Payara shown in *Install Payara* and unzip it to a reasonable location such as `/usr/local/payara7`.
   - Note that Payara can also be downloaded from Maven Central.
   - Note that another way to check the expected version of Payara is to run this command:

     ```
     mvn help:evaluate -Dexpression=payara.version -q -DforceStdout
     ```

2. Install Payara Tools plugin in your IDE:

   Netbeans

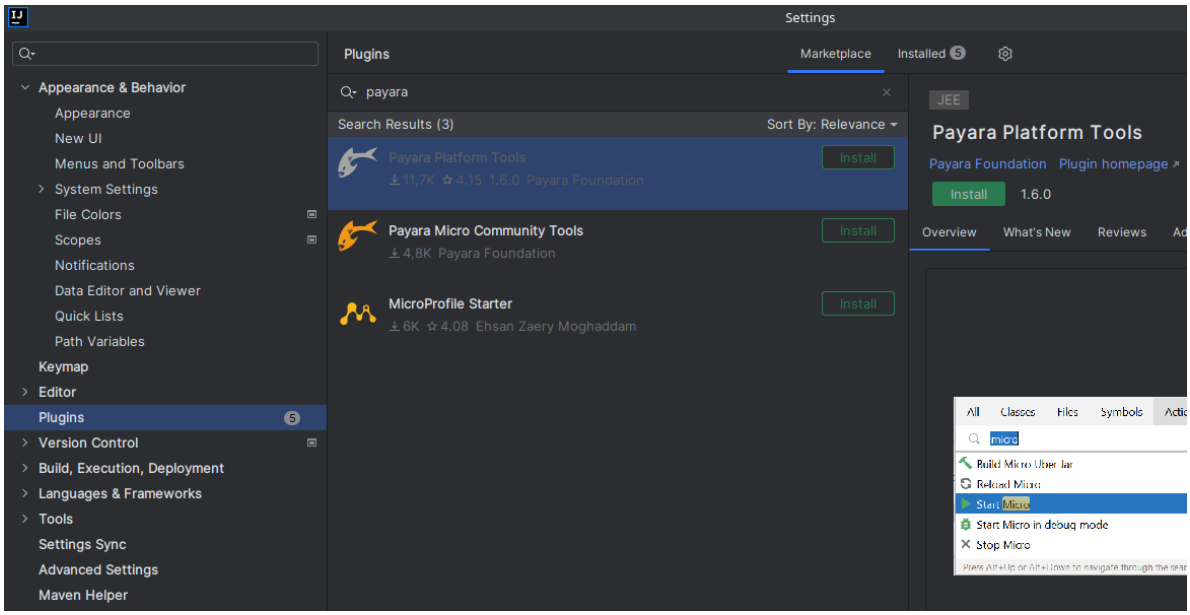
   IntelliJ

   This step is not necessary for Netbeans. The feature is builtin.

   Requires IntelliJ Ultimate! (Note that free educational licenses are available)

   Go to settings, then plugins. Install "Payara Ultimate Tools". For more information:

   - plugin homepage
   - docs
   - source
   - issues



3. Configure a connection to Payara:

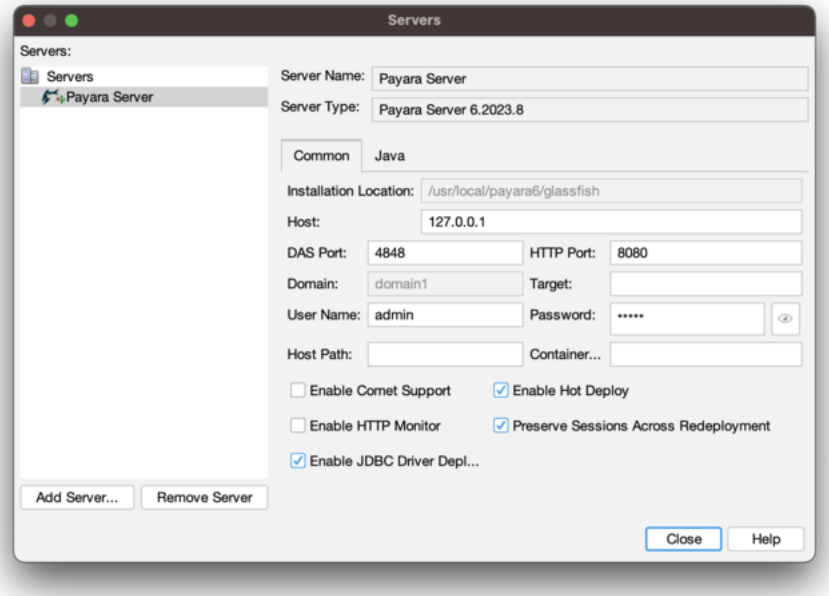
   Netbeans

   IntelliJ

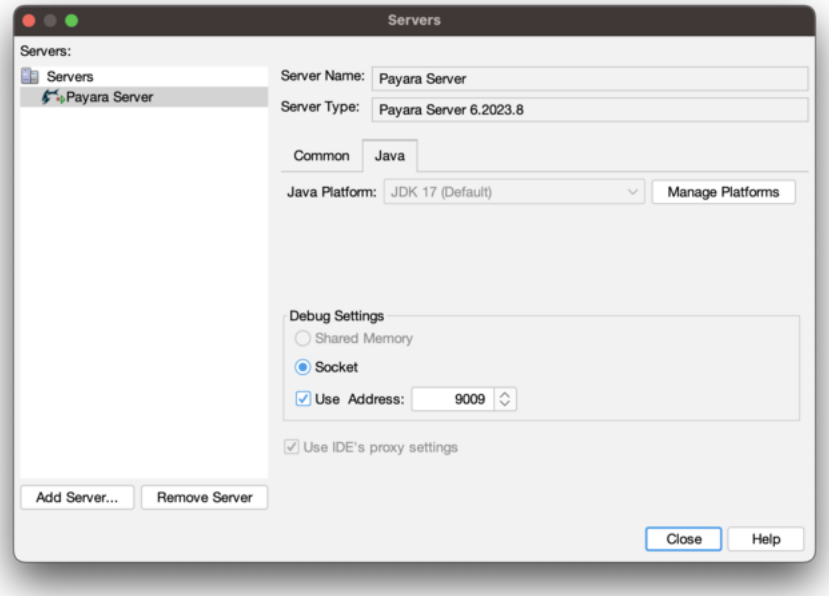
   Launch Netbeans and click "Tools" and then "Servers". Click "Add Server" and select "Payara Server" and set the installation location to `/usr/local/payara7` (or wherever you unzipped Payara). Choose "Remote Domain".

Use the settings in the screenshot below. Most of the defaults are fine.

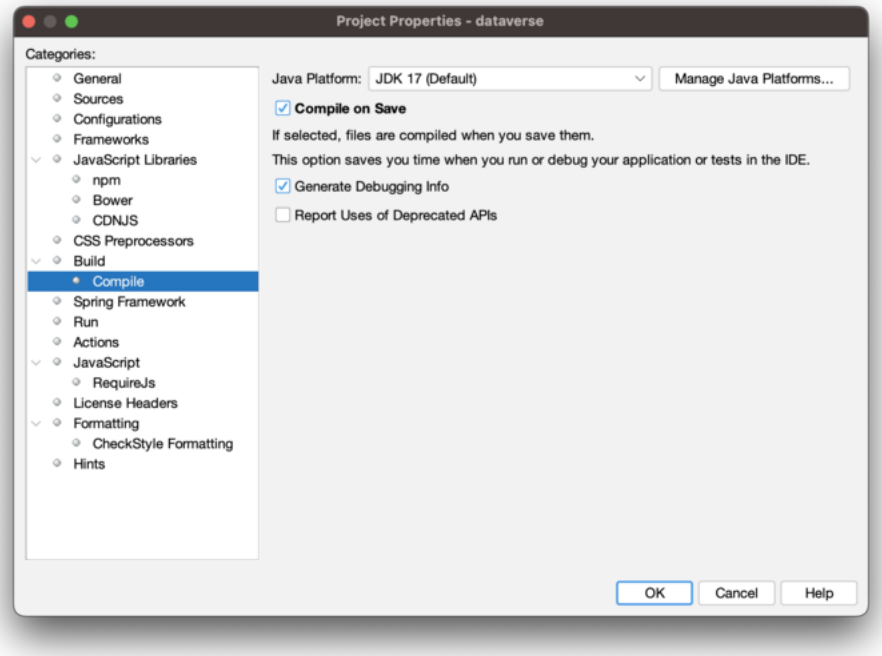
Under "Common", the username and password should be "admin". Make sure "Enable Hot Deploy" is checked.



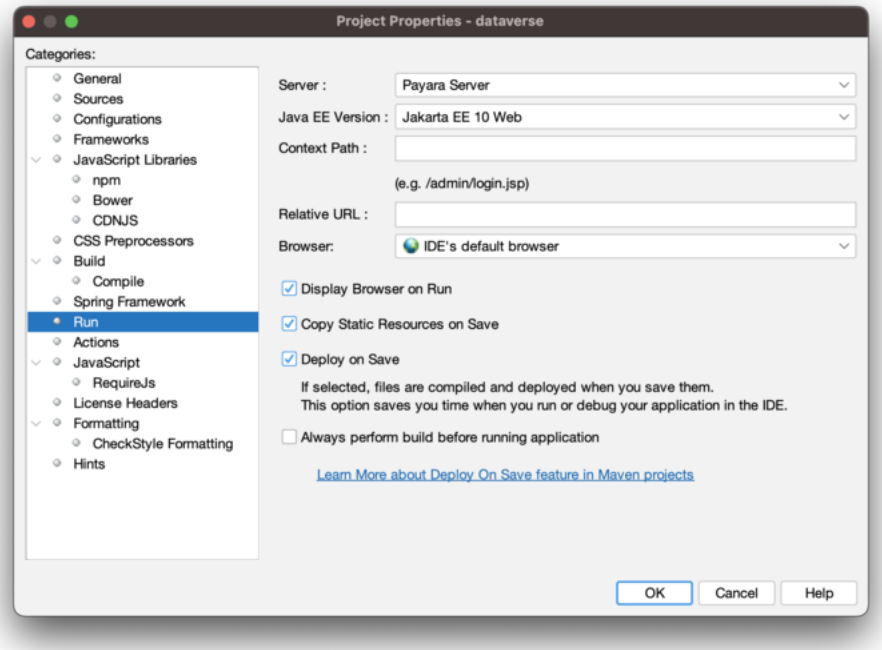
Under "Java", change the debug port to 9009.



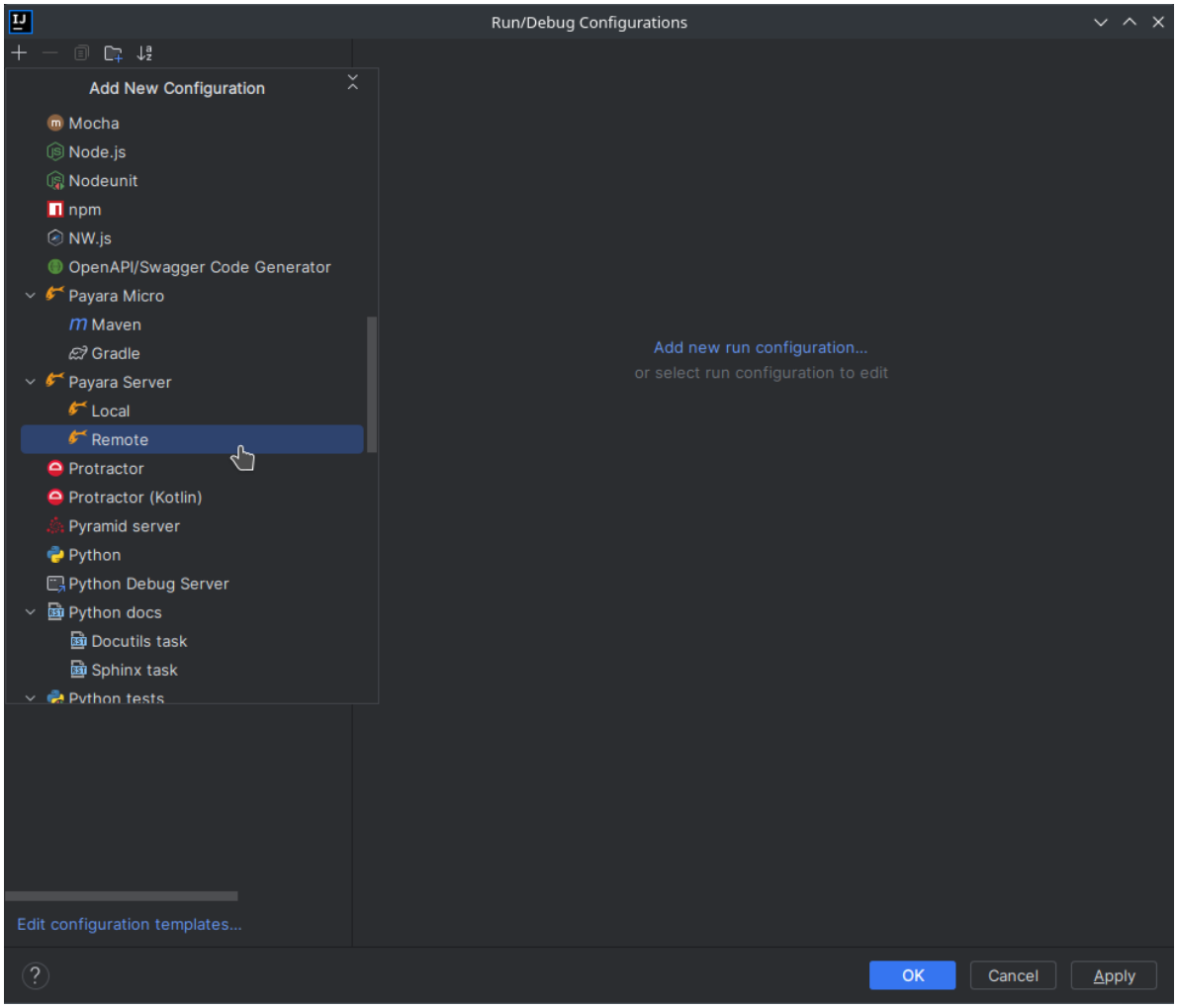
Open the project properties (under "File"), navigate to "Compile" and make sure "Compile on Save" is checked.



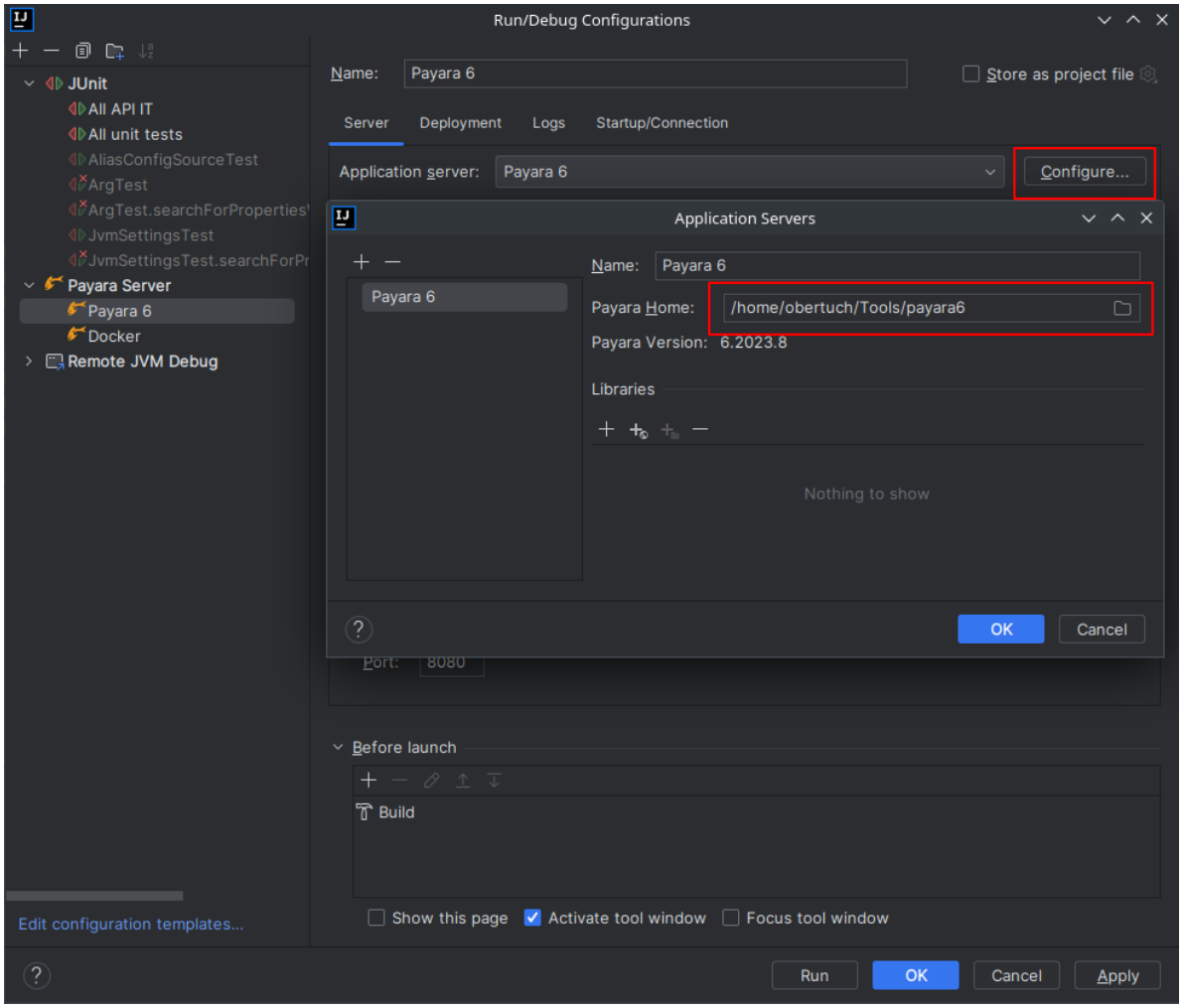
Under “Run”, under “Server”, select “Payara Server”. Make sure “Deploy on Save” is checked.



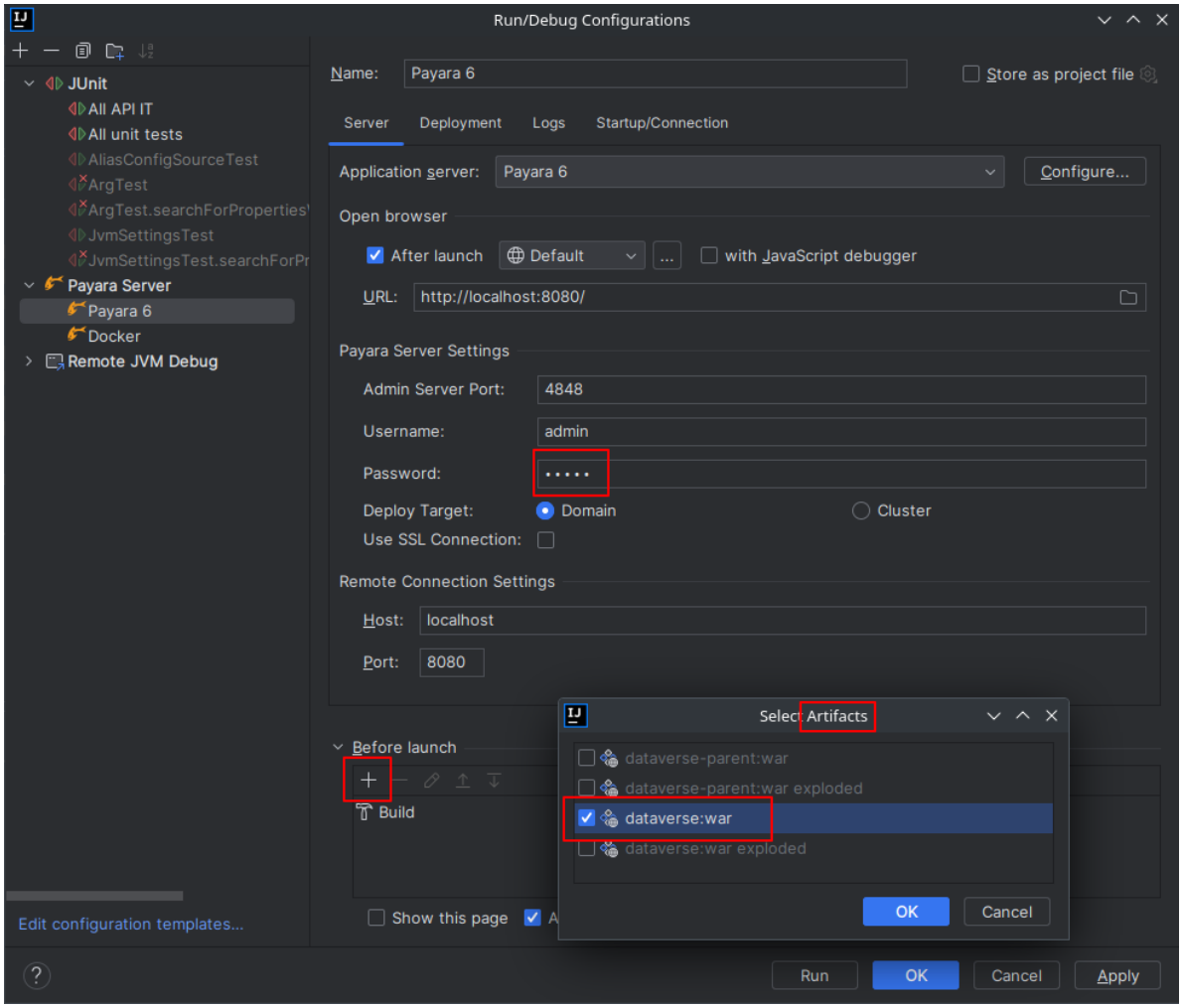
Create a new running configuration with a “Remote Payara”. (Open dialog by clicking “Run”, then “Edit Configurations”)



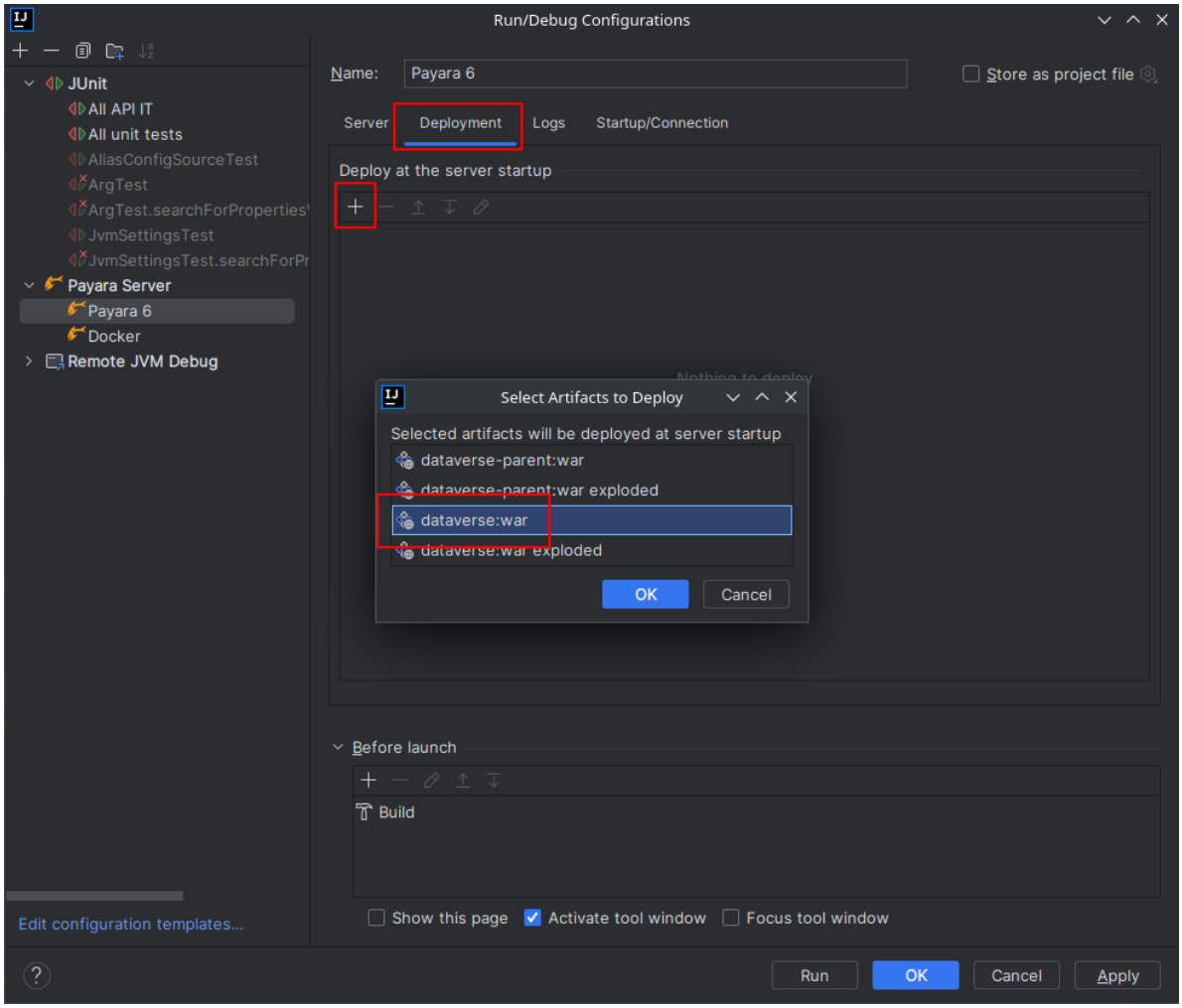
Click on “Configure” next to “Application Server”. Add an application server and select unzipped local directory.



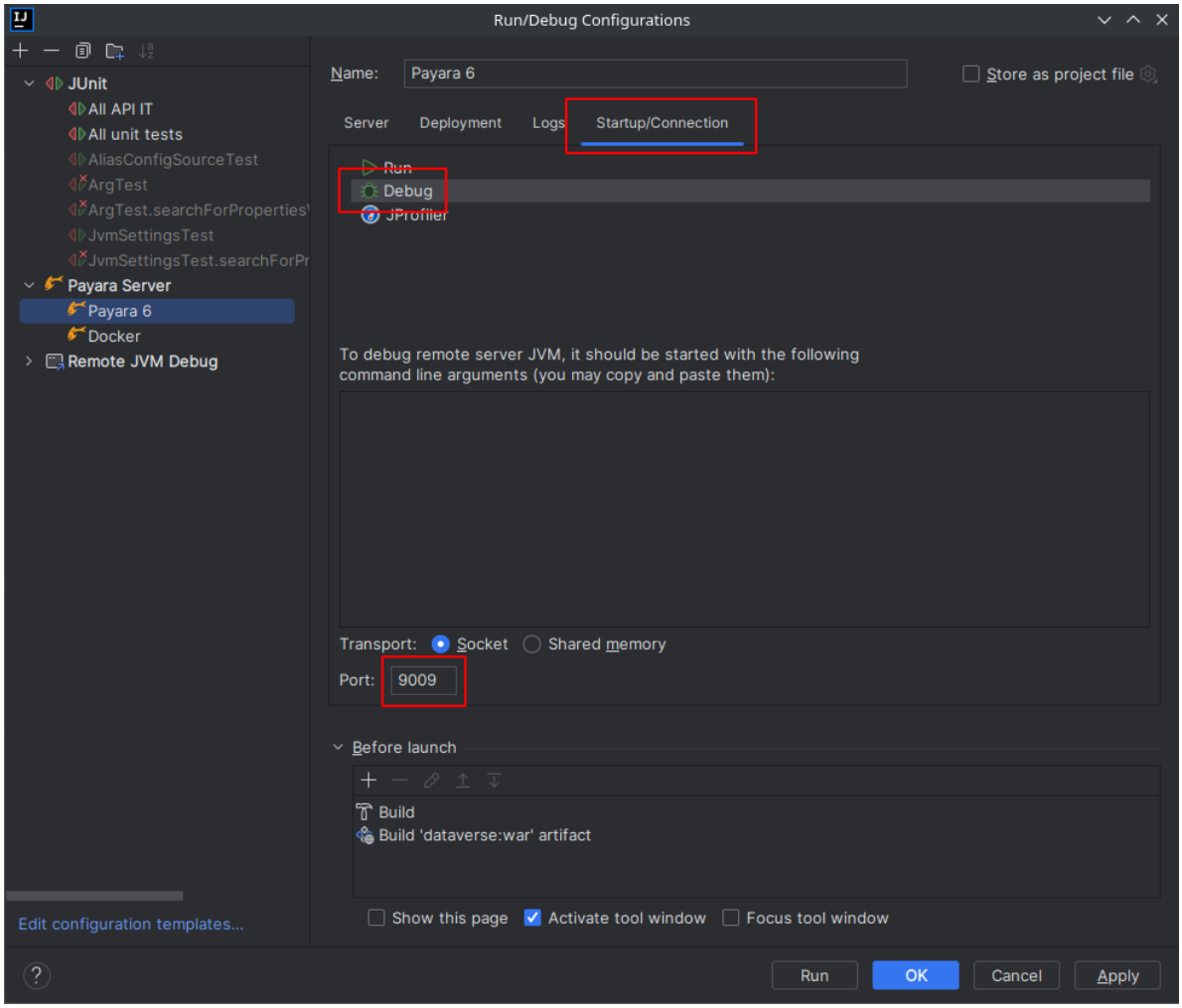
Add admin password “admin” and add “building artifact” before launch. Make sure to select the WAR, *not* exploded!



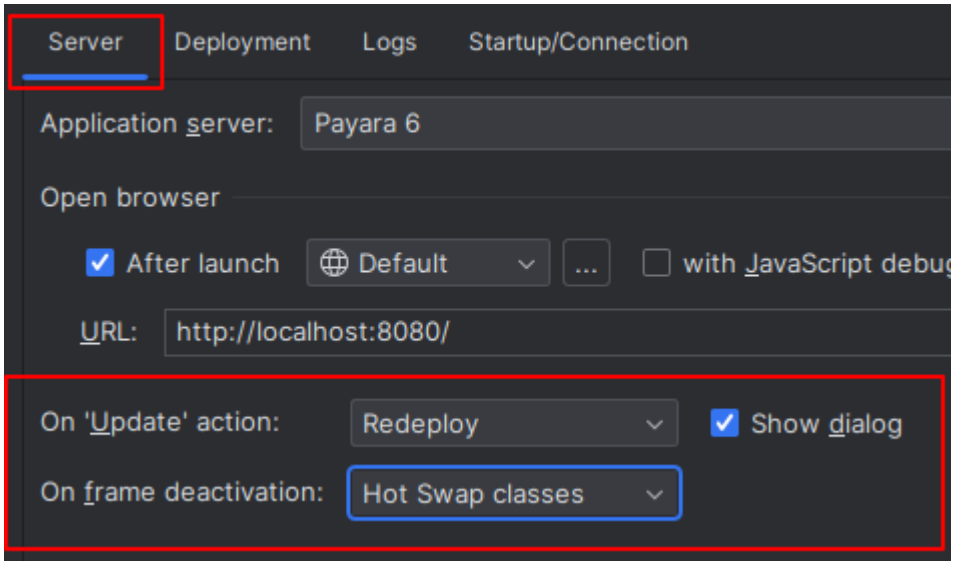
Go to “Deployment” tab and add the Dataverse WAR, *not* exploded!.



Go to “Startup/Connection” tab, select “Debug” and change port to `9009`.



You might want to tweak the hot deploy behavior in the "Server" tab now. "Update action" can be found in the run window (see below). By default it is "Hot Swap classes", which works fine, but as the screenshot shows you can also change it to "Redeploy". "Frame deactivation" means switching from IntelliJ window to something else, e.g. your browser. *Note: static resources like properties, XHTML etc will only update when redeploying!*



4. Start all the containers, but take care to skip application deployment.

   Maven

   Compose

   IntelliJ

   ```
   mvn -Pct docker:run -Dapp.skipDeploy
   ```

   Run above command in your terminal to start containers in foreground and skip deployment. See cheat sheet above for more options. Note that this command either assumes you built the *Application Image* first or will download it from Docker Hub.

   ```
   SKIP_DEPLOY=1 docker compose -f docker-compose-dev.yml up
   ```

   Run above command in your terminal to start containers in foreground and skip deployment. See cheat sheet above for more options. Note that this command either assumes you built the *Application Image* first or will download it from Docker Hub.

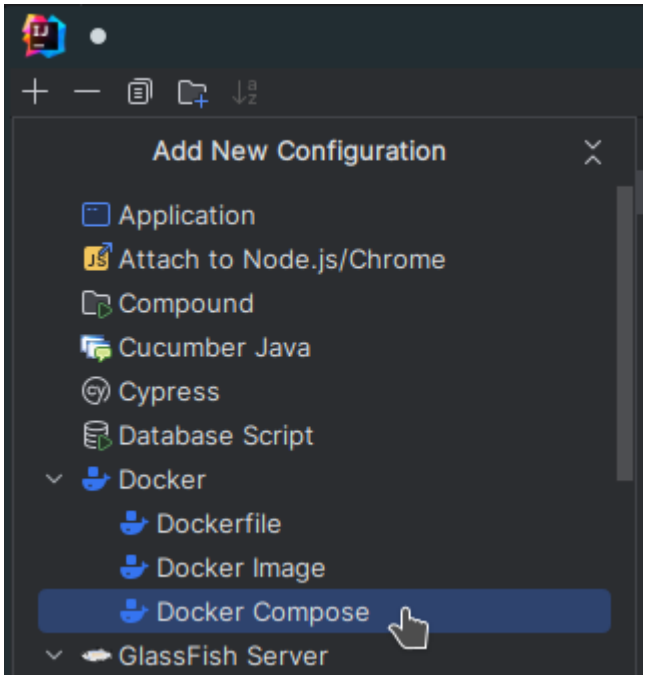
   Note that you can skip this step if you're ok running the command under the "Maven" tab, which is this:

   ```
   mvn -Pct docker:run -Dapp.skipDeploy
   ```

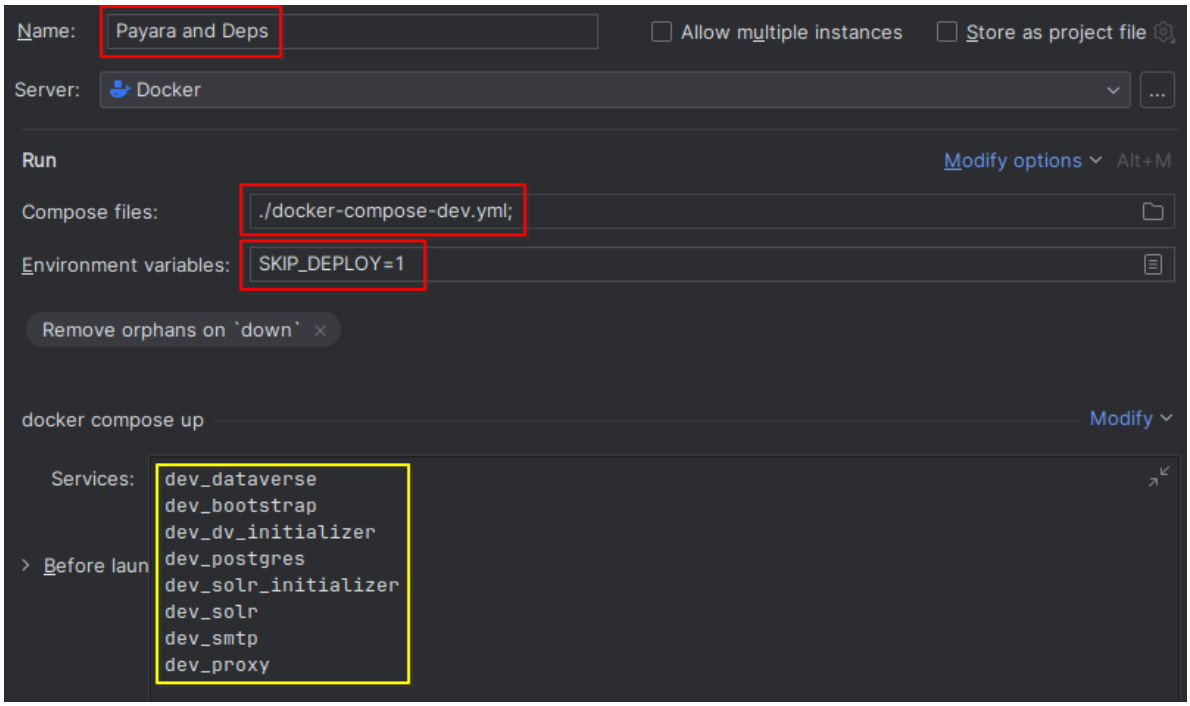
   In IntelliJ you can create a service configuration to automatically start services for you.

   IMPORTANT: This requires installation of the Docker plugin.

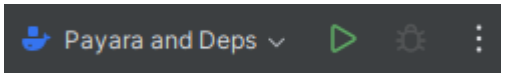
   NOTE: You might need to change the Docker Compose executable in your IDE settings to `docker` if you have no `docker-compose` binary. Start from the `File` menu if you are on Linux/Windows or `IntelliJ IDEA` on Mac and then go to Settings > Build > Docker > Tools.

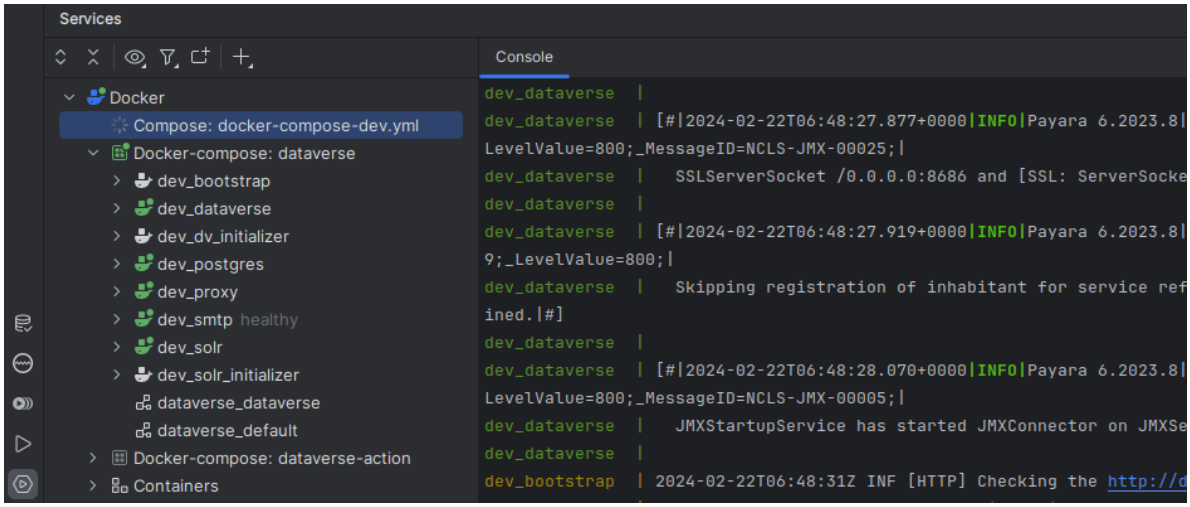


Give your configuration a meaningful name, select the compose file to use (in this case the default one), add the environment variable `SKIP_DEPLOY=1`, and optionally select the services to start. You might also want to change other options like attaching to containers to view the logs within the "Services" tab.



Now run the configuration to prepare for deployment and watch it unfold in the "Services" tab.





Note: the Admin Console can be reached at http://localhost:4848 or https://localhost:4949

5. To deploy the application to the running server, use the configured tools to deploy. Using the "Run" configuration only deploys and enables redeploys, while running "Debug" enables hot swapping of classes via JDWP.

   Netbeans

   IntelliJ

   Click "Debug" then "Debug Project". After some time, Dataverse will be deployed.

   Try making a code change, perhaps to `Info.java`.

   Click "Debug" and then "Apply Code Changes". If the change was correctly applied, you should see output similar to this:

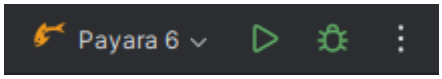
```
Classes to reload:
edu.harvard.iq.dataverse.api.Info

Code updated
```

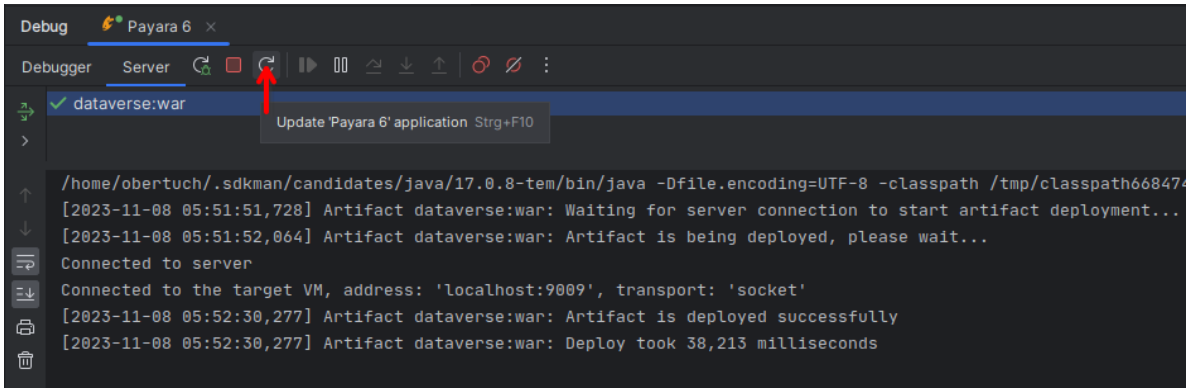
   Check to make sure the change is live by visiting, for example, http://localhost:8080/api/info/version

   See below for a video demonstrating the steps above but please note that the ports used have changed and now that we have the concept of "skip deploy" the undeployment step shown is no longer necessary.

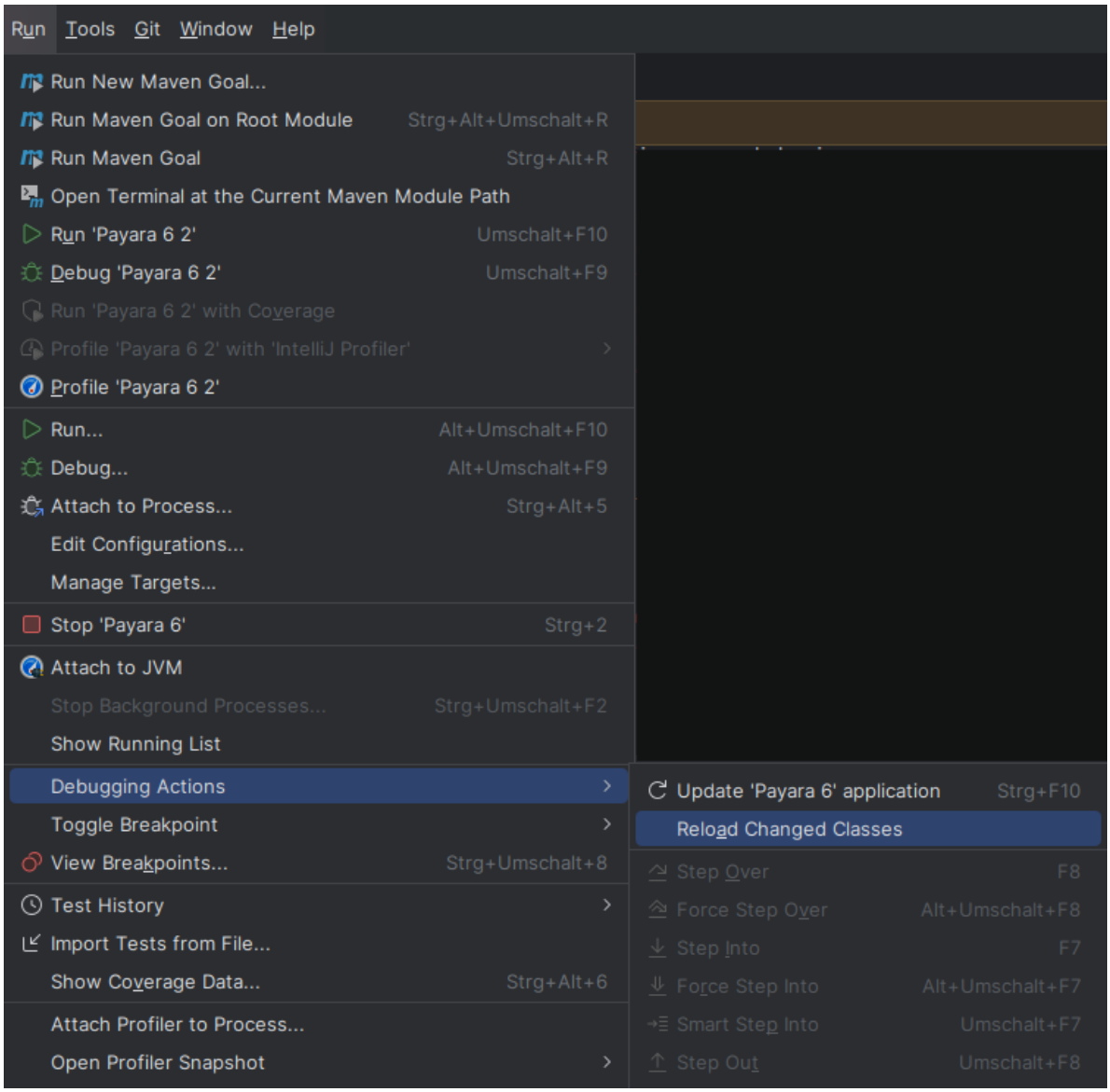
   Choose "Run" or "Debug" in the toolbar.



   Watch the WAR build and the deployment unfold. Note the "Update" action button (see config to change its behavior).



Manually hotswap classes in "Debug" mode via "Run" > "Debugging Actions" > "Compile and Reload Modified Files".



Note: in the background, the bootstrap job will wait for Dataverse to be deployed and responsive. When your IDE automatically opens the URL a newly deployed, not bootstrapped Dataverse application, it might take some more time

and page refreshes until the job finishes.

IDE Triggered Non-Code Re-Deployments

Either redeploy the WAR (see above), use JRebel or look into copying files into the exploded WAR within the running container. The steps below describe options to enable the later in different IDEs.

IntelliJ

This imitates the Netbeans builtin function to copy changes to files under `src/main/webapp` into a destination folder. It is different in the way that it will copy the files into the running container deployment without using a bind mount.

1. Install the File Watchers plugin
2. Import the `watchers.xml` file at *File* > *Settings* > *Tools* > *File Watchers*
3. Once you have the deployment running (see above), editing files under `src/main/webapp` will be copied into the container after saving the edited file. Note: by default, IDE auto-saves will not trigger the copy.
4. Changes are visible once you reload the browser window.

IMPORTANT: This tool assumes you are using the *IDE Triggered Code Re-Deployments* method to run Dataverse.

IMPORTANT: This tool uses a Bash shell script and is thus limited to Mac and Linux OS.

Fast Redeploy (Command-Line)

For developers who prefer command-line workflows over IDE integration, Dataverse provides scripts for fast iterative development without full container rebuilds.

Initial Setup

Run once per development session:

```
./scripts/dev/dev-start-frd.sh
```

This command:

- Builds the full Dataverse WAR with `mvn package`
- Extracts it into `target/dataverse/` as an exploded WAR
- Configures JPA settings for development (`ddl-generation=none`)
- Starts the dev stack with `SKIP_DEPLOY=1`
- Manually deploys the application via `asadmin`

Iterative Development

After making code changes, run:

```
./scripts/dev/dev-frd.sh
```

This script:

- Compiles Java sources incrementally (`mvn compile`, ~5-10s)
- Syncs updated classes and webapp resources into the mounted exploded WAR
- Forces Payara to redeploy the application without restarting containers

- Key features: - Skips full Maven rebuilds (only compiles changed Java files) - Avoids container restarts (uses hot-redeployment) - Completes in ~12 seconds vs. ~54s for traditional full rebuild workflow (4.5x faster) - Preserves database state between deployments

Typical Workflow

```
# Start dev environment once
./scripts/dev/dev-start-frd.sh

# Edit Java or XHTML files...

# Fast redeploy
./scripts/dev/dev-frd.sh

# Repeat as needed

# When finished, stop containers
./scripts/dev/dev-down-frd.sh
```

Memory Configuration

The fast-redeploy workflow includes `docker-compose.override.yml` that increases the memory limit to 8GB (from the default 2GB limit set for GitHub Actions CI) which is insufficient for local Dataverse development. The override file is automatically used by the scripts.

Limitations

- Does not update dependencies (run full `mvn package` + restart if `pom.xml` changes)
- Static resources (CSS, JS) may require browser cache clear
- For database schema changes, use `dev-rebuild.sh` instead
- Performance timings may vary depending on your hardware configuration

Note: This workflow complements IDE-based redeployment. Use whichever fits your development style.

9.3.7 Exploring the Database

See *Database Name and Credentials* in the Developer Guide.

9.3.8 Using a Debugger

The *Base Image* enables usage of the Java Debugging Wire Protocol for remote debugging if you set `ENABLE_JDWP=1` as environment variable for the application container. The default configuration when executing containers with the commands listed at *Running* already enables this.

There are a lot of tutorials how to connect your IDE's debugger to a remote endpoint. Please use `localhost:9009` as the endpoint. Here are links to the most common IDEs docs on remote debugging: Eclipse, IntelliJ

9.3.9 Building Your Own Base Image

If you find yourself tasked with upgrading Payara (or reviewing or testing such an upgrade), you will need to create your own base image before running the *Quickstart*. For instructions, see *Build Instructions* under *Base Image*.

9.4 Base Image

The base image contains Payara and other dependencies that the Dataverse software runs on. It is the foundation for the *Application Image*. Note that some dependencies, such as PostgreSQL and Solr, run in their own containers and are not part of the base image.

Contents:

- *Supported Image Tags*
  - *Tags for Production Use*
  - *Tags for Development Use*
- *Image Contents*
- *Build Instructions*
  - *Automated Builds & Publishing*
  - *Processor Architecture and Multiarch*
- *Tunables*
- *Locations*
- *Exposed Ports*
- *Entry & Extension Points*
- *Other Hints*

A "base image" offers you a pre-installed and pre-tuned application server to deploy Dataverse software to. Adding basic functionality like executing scripts at container boot, monitoring, memory tweaks etc. is all done at this layer, to make the application image focus on the app itself.

NOTE: The base image does not contain the Dataverse application itself.

Within the main repository, you may find the base image's files at `<git root>/modules/container-base`. This Maven module uses the Maven Docker Plugin to build and ship the image. You may use, extend, or alter this image to your liking and/or host in some different registry if you want to.

9.4.1 Supported Image Tags

This image is sourced from the main upstream code repository of the Dataverse software. Development and maintenance of the image's code happens there (again, by the community).

All supported images receive scheduled maintenance, executed every Sunday. New revisions are kept to a minimum, usually created when some dependency needs (security) updates. (Examples: JRE patch releases, ImageMagick fixes, etc.)

Our tagging is inspired by Bitnami and we offer two categories of tags:

- rolling: images change over time
- immutable: images are fixed and never change

In the tags below you'll see the term "flavor". This refers to flavor of Linux the container is built on. We use Ubuntu as the basis for our images and, for the time being, the only operating system flavors we use and support are `noble` (6.4+) and `jammy` (pre-6.4).

You can find all the tags at https://hub.docker.com/r/gdcc/base/tags

Tags for Production Use

The images of the three latest releases of the Dataverse project will receive updates such as security patches for the underlying operating system. Content will be fairly stable as disruptive changes like Payara or Java upgrades will be handled in a new major or minor upgrade to Dataverse (a new `<dv-major>.<dv-minor>` tag). Expect disruptive changes in case of high risk security threats.

- **Latest**
  Definition: `latest`
  Summary: Rolling tag, always pointing to the latest revision of the most current Dataverse release.
- **Rolling Production**
  Definition: `<dv-major>.<dv-minor>-<flavor>`
  Example: `6.12-noble`
  Summary: Rolling tag, pointing to the latest revision of an immutable production image for released versions of Dataverse.
- **Immutable Production**
  Definition: `<dv-major>.<dv-minor>-<flavor>-r<revision>`
  Example: `6.12-noble-r1`
  Summary: An **immutable tag** where the revision is incremented for rebuilds of the image.
  This image should be especially attractive if you want explict control over when your images are updated.

Tags for Development Use

All of the tags below are strongly recommended for development purposes only due to their fast changing nature. In addition to updates due to PR merges, the most recent are undergoing scheduled maintenance to ensure timely security fixes. When a development cycle of the Dataverse project finishes, maintenance ceases for any tags carrying version numbers. For now, stale images will be kept on Docker Hub indefinitely.

- **Unstable**
  Definition: `unstable`
  Summary: Rolling tag, tracking the `develop` branch (see also *The "develop" Branch*). (Dockerfile)
  Please expect abrupt changes like new Payara or Java versions as well as OS updates or flavor switches when using this tag.

- **Upcoming**
  Definition: `<dv-major>.<dv-minor-next>-<flavor>`
  Example: `6.13-noble`
  Summary: Rolling tag, equivalent to `unstable` for current development cycle. Will roll over to the rolling production tag after a Dataverse release.
- **Flexible Stack**
  Definition: `<dv-major>.<dv-minor-next>-<flavor>-p<payara.version>-j<java.version>`
  Example: `6.13-noble-p6.2025.3-j17`
  Summary: Rolling tag during a development cycle of the Dataverse software (Dockerfile).

NOTE: In these tags for development usage, the version number will always be 1 minor version ahead of existing Dataverse releases. Example: Assume Dataverse `6.x` is released, `6.(x+1)` is underway. The rolling tag in use during the cycle will be `6.(x+1)-FFF` and `6.(x+1)-FFF-p6.202P.P-jJJ`. See also: *Making Releases*.

9.4.2 Image Contents

The base image provides:

- Eclipse Temurin JRE
- Payara Community Application Server
- CLI tools necessary to run Dataverse (i. e. `curl` or `jq` - see also *Prerequisites* in Installation Guide)
- Linux tools for analysis, monitoring and so on
- Jattach (attach to running JVM)
- wait4x (tool to "wait for" a service to be available)
- dumb-init (see *below* for details)

This image is created as a "multi-arch image", see *below*.

It inherits (is built on) an Ubuntu environment from the upstream base image of Eclipse Temurin. You are free to change the JRE/JDK image to your liking (see below).

9.4.3 Build Instructions

Assuming you have Docker, Docker Desktop, Moby or some remote Docker host configured, up and running from here on.

Simply execute the Maven modules packaging target with activated "container" profile. Either from the projects Git root:

```
mvn -Pct -f modules/container-base install
```

Or move to the module and execute:

```
cd modules/container-base && mvn -Pct install
```

Some additional notes, using Maven parameters to change the build and use ...:

- ... a different tag only: add `-Dbase.image.tag=tag`.
  Note: default is `unstable`
- ... a different image name and tag: add `-Dbase.image=name:tag`.
  Note: default is `gdcc/base:${base.image.tag}`

- … a different image registry than Docker Hub: add `-Ddocker.registry=registry.example.org` (see also DMP docs on registries)
- … a different Payara version: add `-Dpayara.version=V.YYYY.R`.
- … a different Temurin JRE version `A`: add `-Dtarget.java.version=A` (i.e. `11`, `17`, …).
  Note: must resolve to an available image tag `A-jre` of Eclipse Temurin! (See also Docker Hub search example)
- … a different Java Distribution: add `-Djava.image="name:tag"` with precise reference to an image available local or remote.
- … a different UID/GID for the `payara` user/group (default `1000:1000`): add `-Dbase.image.uid=1234` (or `.gid`)

Automated Builds & Publishing

To make reusing most simple, the image is built with a Github Action within the IQSS repository and then pushed to Docker Hub gdcc/base repository. It is built and pushed on every edit to its sources plus uncached scheduled nightly builds to make sure security updates are finding their way in.

Note: For the Github Action to be able to push to Docker Hub, two repository secrets (DOCKERHUB_USERNAME, DOCKERHUB_TOKEN) have been added by IQSS admins to their repository.

Processor Architecture and Multiarch

This image is created as a "multi-arch image", supporting the most common architectures Dataverse usually runs on: AMD64 (Windows/Linux/…) and ARM64 (Apple M1/M2), by using Maven Docker Plugin's BuildX mode.

Building the image via `mvn -Pct package` or `mvn -Pct install` as above will only build for the architecture of the Docker machine's CPU.

Only `mvn -Pct deploy` will trigger building on all enabled architectures (and will try to push the images to a registry, which is Docker Hub by default).

You can specify which architectures you would like to build for and include by them as a comma separated list: `mvn -Pct deploy -Ddocker.platforms="linux/amd64,linux/arm64"`. The shown configuration is the default and may be omitted.

Yet, to enable building with non-native code on your build machine, you will need to setup a cross-platform builder!

On Linux, you should install qemu-user-static (preferably via your package management) on the host and run `docker run --rm --privileged multiarch/qemu-user-static --reset -p yes` to enable that builder. The Docker plugin will setup everything else for you.

The upstream CI workflows publish images supporting AMD64 and ARM64 (see e.g. tag details on Docker Hub)

9.4.4 Tunables

The base image provides a Payara domain suited for production use, but can also be used during development. Many settings have been carefully selected for best performance and stability of the Dataverse application.

As with any service, you should always monitor any metrics and make use of the tuning capabilities the base image provides. These are mostly based on environment variables (very common with containers) and provide sane defaults.

Env. variable	Default	Type	Description
DEPLOY_PRO	(empty)	String	Set to add arguments to generated *asadmin deploy* commands.
PREBOOT_CO	[preboot]	Abs. path	Provide path to file with `asadmin` commands to run **before** boot of application server. See also Pre/postboot script docs. Must be writeable by Payara Linux user!
POSTBOOT_C	[postboot]	Abs. path	Provide path to file with `asadmin` commands to run **after** boot of application server. See also Pre/postboot script docs. Must be writeable by Payara Linux user!
JVM_ARGS	(empty)	String	Additional arguments to pass to application server's JVM on start.
MEM_MAX_RA	`70.0`	Percentage	Maximum amount of container's allocated RAM to be used as heap space. Make sure to leave some room for native memory, OS overhead etc!
MEM_XSS	`512k`	Size	Tune the maximum JVM stack size.
MEM_MIN_HE	`20`	Integer	Make the heap shrink aggressively and grow conservatively. See also run-java-sh recommendations.
MEM_MAX_HE	`40`	Integer	Make the heap shrink aggressively and grow conservatively. See also run-java-sh recommendations.
MEM_MAX_GC	`500`	Milliseconds	Shorter pause times might result in lots of collections causing overhead without much gain. This needs monitoring and tuning. It's a complex matter.
MEM_METASP	`256m`	Size	Initial size of memory reserved for class metadata, also used as trigger to run a garbage collection once passing this size.
MEM_MAX_ME	`2g`	Size	The metaspace's size will not outgrow this limit.
ENABLE_DUM	`0`	Bool, `0|1`	If enabled, the argument(s) given in `JVM_DUMP_ARG` will be added to the JVM starting up. This means it will enable dumping the heap to `${DUMPS_DIR}` (see below) in "out of memory" cases. (You should back this location with disk space / ramdisk, so it does not write into an overlay filesystem!)
JVM_DUMPS_	[dump-optior	String	Can be fine tuned for more grained controls of dumping behaviour.
ENABLE_JMX	`0`	Bool, `0|1`	Allow insecure JMX connections, enable AMX and tune all JMX monitoring levels to `HIGH`. See also Payara Docs - Basic Monitoring. A basic JMX service is enabled by default in Payara, exposing basic JVM MBeans, but especially no Payara MBeans.
ENABLE_JDW	`0`	Bool, `0|1`	Enable the "Java Debug Wire Protocol" to attach a remote debugger to the JVM in this container. Listens on port 9009 when enabled. Search the internet for numerous tutorials to use it.
ENABLE_REL	`0`	Bool, `0|1`	Enable the dynamic "hot" reloads of files when changed in a deployment. Useful for development, when new artifacts are copied into the running domain. Also, export Dataverse specific environment variables `DATAVERSE_JSF_PROJECT_STAGE=Development` and `DATAVERSE_JSF_REFRESH_PERIOD=0` to enable dynamic JSF page reloads.
SKIP_DEPLO	`0`	Bool, `0|1` or `false|true`	When active, do not deploy applications from `DEPLOY_DIR` (see below), just start the application server. Will still execute any provided init scripts and only skip deployments within the default init scripts.
DATAVERSE_	`900`	Seconds	See *Application Server Settings* `http.request-timeout-seconds`. *Note:* can also be set using any other MicroProfile Config Sources available via `dataverse.http.timeout`.
PAYARA_ADM	`admin`	String	Set to secret string to change Payara Admin Console Adminstrator User ("admin") password.
LINUX_PASS	`payara`	String	Set to secret string to change the Payara Linux User ("payara", default UID=1000) password. *Note: changes /etc/shadow, usually incompatible with a read-only rootfs!*
DOMAIN_PAS	`changeit`	String	Set to secret string to change the Domain Master Password.

9.4.5 Locations

This environment variables represent certain locations and might be reused in your scripts etc. All of these variables aren't meant to be reconfigurable and reflect state in the filesystem layout!

Writeable at build time:

The overlay filesystem of Docker and other container technologies is not meant to be used for any performance IO. You should avoid *writing* data anywhere in the file tree at runtime, except for well known locations with mounted volumes backing them (see below).

The locations below are meant to be written to when you build a container image, either this base or anything building upon it. You can also use these for references in scripts, etc.

Env. variable	Value	Description
`HOME_DIR`	`/opt/payara`	Home base to Payara and the application
`PAYARA_DIR`	`${HOME_DIR}/appserver`	Installation directory of Payara server
`SCRIPT_DIR`	`${HOME_DIR}/scripts`	Any scripts like the container entrypoint, init scripts, etc
`CONFIG_DIR`	`${HOME_DIR}/config`	Payara Server configurations like pre/postboot command files go here (Might be reused for Dataverse one day)
`DEPLOY_DIR`	`${HOME_DIR}/deployments`	Any EAR or WAR file, exploded WAR directory etc are autodeployed on start. See also `SKIP_DEPLOY` above.
`DOMAIN_DIR`	`${PAYARA_DIR}/glassfish/domains/${DOMAIN_NAME}`	Path to root of the Payara domain applications will be deployed into. Usually `${DOMAIN_NAME}` will be `domain1`.

Writeable at runtime:

The locations below are defined as Docker volumes by the base image. They will by default get backed by an "anonymous volume", but you can (and should) bind-mount a host directory or named Docker volume in these places to avoid data loss, gain performance and/or use a network file system.

Notes: 1. On Kubernetes you still need to provide volume definitions for these places in your deployment objects! 2. You should not write data into these locations at build time - it will be shadowed by the mounted volumes!

Env. variable	Value	Description
`STORAGE_DIR`	`/dv`	This place is writeable by the Payara user, making it usable as a place to store research data, customizations or other. Images inheriting the base image should create distinct folders here, backed by different mounted volumes. Enforce correct filesystem permissions on the mounted volume using `fix-fs-perms.sh` from *Config Baker Image* or similar scripts.
`SECRETS_DIR`	`/secrets`	Mount secrets or other here, being picked up automatically by Directory Config Source. See also various *Configuration* options involving secrets.
`DUMPS_DIR`	`/dumps`	Default location where heap dumps will be stored (see above). You should mount some storage here (disk or ephemeral).

9.4.6 Exposed Ports

The default ports that are exposed by this image are:

- 8080 - HTTP listener
- 4848 - Admin Service HTTPS listener
- 8686 - JMX listener
- 9009 - "Java Debug Wire Protocol" port (when `ENABLE_JDWP=1`)

The HTTPS listener (on port 8181) becomes deactivated during the build, as we will always need to reverse-proxy the application server and handle SSL/TLS termination at this point. Save the memory and some CPU cycles!

9.4.7 Entry & Extension Points

The entrypoint shell script provided by this base image will by default ensure to:

- Run any scripts named `${SCRIPT_DIR}/init_*` or in `${SCRIPT_DIR}/init.d/*` directory for initialization **before** the application server starts.
- Run an executable script `${SCRIPT_DIR}/startInBackground.sh` in the background - if present.
- Run the application server startup scripting in foreground (`${SCRIPT_DIR}/startInForeground.sh`).

If you need to create some scripting that runs in parallel under supervision of dumb-init, e.g. to wait for the application to deploy before executing something, this is your point of extension: simply provide the `${SCRIPT_DIR}/startInBackground.sh` executable script with your application image.

9.4.8 Other Hints

By default, `domain1` is enabled to use the `G1GC` garbage collector.

To access the Payara Admin Console or use the `asadmin` command, use username `admin` and password `admin`.

For running a Java application within a Linux based container, the support for CGroups is essential. It has been included and activated by default since Java 8u192, Java 11 LTS and later. If you are interested in more details, you can read about those in a few places like https://developers.redhat.com/articles/2022/04/19/java-17-whats-new-openjdks-container-awareness, https://www.eclipse.org/openj9/docs/xxusecontainersupport, etc. The other memory defaults are inspired from run-java-sh recommendations.

9.5 Application Image

The application image is a layer on top of the base image and contains the Dataverse software.

Contents:

- *Supported Image Tags*
  - *Tags for Production Use*
  - *Tags for Development Use*
- *Image Contents*

- *Build Instructions*
  - *Automated Builds & Publishing*
  - *Processor Architecture and Multiarch*
- *Tunables*
- *Locations*
- *Exposed Ports*
- *Entry & Extension Points*

An "application image" offers you a deployment ready Dataverse application running on the underlying application server, which is provided by the *Base Image*. Its sole purpose is to bundle the application and any additional material necessary to successfully jumpstart the application.

Until all *JVM Options* are *MicroProfile Config* enabled, it also adds the necessary scripting glue to configure the applications domain during booting the application server. See *Tunables*.

Within the main repository, you may find the application image's files at `<git root>/src/main/docker`. This is the same Maven module providing a Dataverse WAR file for classic installations, and uses the Maven Docker Plugin to build and ship the image within a special Maven profile.

9.5.1 Supported Image Tags

This image is sourced from the main upstream code repository of the Dataverse software. Development and maintenance of the image's code happens there (again, by the community).

All supported images receive scheduled maintenance, executed every Sunday. New revisions are kept to a minimum, usually created when some dependency needs (security) updates. For the application images it correlates mostly to the *base image* receiving updates.

Our tagging is inspired by Bitnami and we offer two categories of tags:

- rolling: images change over time
- immutable: images are fixed and never change

In the tags below you'll see the term "flavor". This refers to flavor of Linux the container is built on. We use Ubuntu as the basis for our images and, for the time being, the only operating system flavors we use and support are `noble` (6.4+) and `jammy` (pre-6.4).

You can find all the tags at https://hub.docker.com/r/gdcc/dataverse/tags

Tags for Production Use

The images of the three latest releases of Dataverse will receive updates such as security patches for the underlying operating system. Content will be fairly stable as disruptive changes like Payara or Java upgrades will be handled in a new major or minor upgrade to Dataverse (a new `<dv-major>.<dv-minor>` tag). Expect disruptive changes in case of high risk security threats.

- **Latest**
  Definition: `latest`
  Summary: Rolling tag, always pointing to the latest revision of the most current Dataverse release. In Dataverse 6.6 and lower, the equivalent was the `alpha` tag, which has been deleted.

- **Rolling Production**
  Definition: `<dv-major>.<dv-minor>-<flavor>`
  Example: `6.12-noble`
  Summary: Rolling tag, pointing to the latest revision of an immutable production image for released versions of Dataverse.
- **Immutable Production**
  Definition: `<dv-major>.<dv-minor>-<flavor>-r<revision>`
  Example: `6.12-noble-r1`
  Summary: An **immutable tag** where the revision is incremented for rebuilds of the image.
  This image should be especially attractive if you want explict control over when your images are updated.

Tags for Development Use

All of the tags below are strongly recommended only for development purposes due to their fast-changing nature. In addition to updates due to PR merges, the most recent tags undergo scheduled maintenance to ensure timely security fixes. When a development cycle of Dataverse finishes (see *Making Releases*), maintenance ceases for any of the tags below carrying version numbers. For now, stale images will be kept on Docker Hub indefinitely.

- **Unstable**
  Definition: `unstable`
  Summary: Rolling tag, tracking the `develop` branch (see also *The "develop" Branch*). (Dockerfile)
  Please expect abrupt changes like new Payara or Java versions as well as OS updates or flavor switches when using this tag.
- **Upcoming**
  Definition: `<dv-major>.<dv-minor-next>-<flavor>`
  Example: `6.13-noble`
  Summary: Rolling tag, equivalent to `unstable` for current development cycle. Will roll over to the rolling production tag after a Dataverse release.
  Discussion: Perhaps you are eager to start testing features of an upcoming version (e.g. 6.13) in a staging environment. You select the `6.13-noble` tag (as opposed to `unstable`) because you want to stay on 6.13 rather than switching to the version **after that** when a release is made (which would happen if you had selected the `unstable` tag). Also, when the next release comes out (6.13 in this example), you would stay on the `6.13-noble` tag, which is the same tag that someone would use who wants the final release of 6.13. (See "Rolling Production", above.)

NOTE: In these tags for development usage, the version number will always be 1 minor version ahead of existing Dataverse releases. Example: Assume Dataverse `6.x` is released, `6.(x+1)` is underway. The rolling tag in use during the cycle will be `6.(x+1)-FFF` and `6.(x+1)-FFF-p6.202P.P-jJJ`. See also: *Making Releases*.

9.5.2 Image Contents

The application image builds by convention upon the *base image* and provides:

- Dataverse class files
- Resource files
- Dependency JAR files
- JHove configuration
- Script to configure the application server domain for *JVM Options* not yet *MicroProfile Config* enabled.

The image is provided as a multi-arch image to support the most common architectures Dataverse usually runs on: AMD64 (Windows/Linux/...) and ARM64 (Apple M1/M2). (Easy to extend.)

9.5.3 Build Instructions

Assuming you have Docker, Docker Desktop, Moby or some remote Docker host configured, up and running from here on.

Simply execute the Maven modules packaging target with activated "container" profile from the projects Git root to compile the Java code and build the image:

```
mvn -Pct clean package
```

Some additional notes, using Maven parameters to change the build and use ...:

- ... a different tag only: add `-Dapp.image.tag=tag`.
  Note: default is `unstable`
- ... a different image name and tag: add `-Dapp.image=name:tag`.
  Note: default is `gdcc/dataverse:${app.image.tag}`
- ... a different image registry than Docker Hub: add `-Ddocker.registry=registry.example.org` (see also DMP docs on registries)
- ... a different base image tag: add `-Dbase.image.tag=tag`
  Note: default is `unstable`
- ... a different base image: add `-Dbase.image=name:tag`
  Note: default is `gdcc/base:${base.image.tag}`. See also *Base Image* for more details on it.

Automated Builds & Publishing

See note above at "Supported Image Tags".

Processor Architecture and Multiarch

This image is created as a "multi-arch image", supporting the most common architectures Dataverse usually runs on: AMD64 (Windows/Linux/...) and ARM64 (Apple M1/M2), by using Maven Docker Plugin's BuildX mode.

Building the image via `mvn -Pct package` or `mvn -Pct install` as above will only build for the architecture of the Docker machine's CPU.

Only `mvn -Pct clean deploy -Ddocker.platforms=linux/amd64,linux/arm64` will trigger building on all enabled architectures. Yet, to enable building with non-native code on your build machine, you will need to setup a cross-platform builder.

On Linux, you should install qemu-user-static (preferably via your package management) on the host and run `docker run --rm --privileged multiarch/qemu-user-static --reset -p yes` to enable that builder. The Docker plugin will setup everything else for you.

9.5.4 Tunables

The *Base Image* provides a long list of possible options to tune many aspects of the application server, and, as the application image builds upon it, *Base Image Tunables* apply to it as well.

In addition, the application image provides the following tunables:

Env. variable	Default	Type	Description
`MP_CONFIG_`	`ct`	String	Set to switch the activated *MicroProfile Config Profile*. Note that certain defaults will not apply any longer. See *Application Server Settings* for details.
`dataverse_` and `doi_*`	-	String	Configure any *JVM Options* not yet *MicroProfile Config* enabled with this magic trick. 1. Simply pick a JVM option from the list and replace any `.` with `_`. 2. Replace any `-` in the option name with `__`.

Note that the script `init_2_configure.sh` will apply a few very important defaults to enable quick usage by a) activating the scheduled tasks timer, b) add local file storage if not disabled, and c) a sensible password reset timeout:

```
dataverse_auth_password__reset__timeout__in__minutes=60
dataverse_timerServer=true
dataverse_files_storage__driver__id=local

if dataverse_files_storage__driver__id = "local" then
    dataverse_files_local_type=file
    dataverse_files_local_label=Local
    dataverse_files_local_directory=${STORAGE_DIR}/store
```

9.5.5 Locations

There are only a few important additions to the list of locations by the base image. Please make sure to back these locations with volumes or tmpfs to avoid writing data into the overlay filesystem, which will significantly hurt performance.

Location	Value	Description
`${STORAGE_DI`	`/dv`	Defined by base image. Either back this folder or, if suitable, the locations below it with volumes or tmpfs.
`${STORAGE_DI` `uploads`	`/dv/uploads`	See *dataverse.files.uploads* for a detailed description.
`${STORAGE_DI` `temp`	`/dv/temp`	See *dataverse.files.directory* for a detailed description.
`${STORAGE_DI` `store`	`/dv/store`	Important when using the default provided local storage option (see above and *File Storage*)
`/tmp`	-	Location for temporary files, see also *Temporary Upload File Storage*

9.5.6 Exposed Ports

See base image *exposed port*.

9.5.7 Entry & Extension Points

The application image makes use of the base image provided system to execute scripts on boot, see *Entry & Extension Points*. See there for potential extension of this image in your own derivative.

9.6 Config Baker Image

The config baker container may be used to execute all sorts of tasks around setting up, preparing and finalizing an instance of the Dataverse software. Its focus is bootstrapping non-initialized installations.

Contents:

- *Quickstart*
- *Supported Image Tags*
  - *Tags for Production Use*
  - *Tags for Development Use*
- *Image Contents*
  - *Scripts*
  - *Solr Template*
- *Build Instructions*
  - *Processor Architecture and Multiarch*
- *Tunables*
- *Locations*
- *Exposed Ports*
- *Entry & Extension Points*
- *Examples*

9.6.1 Quickstart

To see the Config Baker help screen:

```
docker run -it --rm gdcc/configbaker:unstable
```

9.6.2 Supported Image Tags

This image is sourced from the main upstream code repository of the Dataverse software. Development and maintenance of the image's code happens there (again, by the community).

All supported images receive scheduled maintenance, executed every Sunday. New revisions are kept to a minimum, usually created when some dependency needs (security) updates. The Trivy scanner is used to check for fixed vulnerabilities and rebuilds are issued if such a fix is detected.

Our tagging is inspired by Bitnami and we offer two categories of tags:

- rolling: images change over time
- immutable: images are fixed and never change

In the tags below you'll see the term "flavor". This refers to flavor of Linux the container is built on. We use Ubuntu as the basis for our images and, for the time being, the only operating system flavors we use and support are `noble` (6.7+) and `alpine` (pre-6.7).

You can find all the tags at https://hub.docker.com/r/gdcc/configbaker/tags

Tags for Production Use

The images of the three latest releases of the Dataverse project will receive updates such as security patches for the underlying operating system. Content will be fairly stable as disruptive changes like Payara or Java upgrades will be handled in a new major or minor upgrade to Dataverse (a new `<dv-major>.<dv-minor>` tag). Expect disruptive changes in case of high risk security threats.

- **Latest**
  Definition: `latest`
  Summary: Rolling tag, always pointing to the latest revision of the most current Dataverse release.
- **Rolling Production**
  Definition: `<dv-major>.<dv-minor>-<flavor>`
  Example: `6.12-noble`
  Summary: Rolling tag, pointing to the latest revision of an immutable production image for released versions of Dataverse.
- **Immutable Production**
  Definition: `<dv-major>.<dv-minor>-<flavor>-r<revision>`
  Example: `6.12-noble-r1`
  Summary: An **immutable tag** where the revision is incremented for rebuilds of the image.
  This image should be especially attractive if you want explict control over when your images are updated.

Tags for Development Use

All of the tags below are strongly recommended only for development purposes due to their fast-changing nature. In addition to updates due to PR merges, the most recent tags undergo scheduled maintenance to ensure timely security fixes. When a development cycle of Dataverse finishes, maintenance ceases for any tags carrying version numbers. For now, stale images will be kept on Docker Hub indefinitely.

- **Unstable**
  Definition: `unstable`
  Summary: Rolling tag, tracking the `develop` branch (see also *The "develop" Branch*). (Dockerfile)
  Please expect abrupt changes like new Payara or Java versions as well as OS updates or flavor switches when using this tag.

- **Upcoming**
  Definition: `<dv-major>.<dv-minor-next>-<flavor>`
  Example: `6.13-noble`
  Summary: Rolling tag, equivalent to `unstable` for current development cycle. Will roll over to the rolling production tag after a Dataverse release.

NOTE: In these tags for development usage, the version number will always be 1 minor version ahead of existing Dataverse releases. Example: Assume Dataverse `6.x` is released, `6.(x+1)` is underway. The rolling tag in use during the cycle will be `6.(x+1)-FFF` and `6.(x+1)-FFF-p6.202P.P-jJJ`. See also: *Making Releases*.

9.6.3 Image Contents

This image contains some crucial parts to make a freshly baked Dataverse installation usable.

Scripts

Script	Description
`bootstrap.sh`	Run an initialization script contained in a persona. See `bootstrap.sh -h` for usage details. For development purposes, use `bootstrap.sh dev` or provide your own.
`fix-fs-perms.sh`	Fixes filesystem permissions. App and Solr container run as non-privileged users and might need adjusted filesystem permissions on mounted volumes to be able to write data. Run without parameters to see usage details.
`help.sh`	Default script when running container without parameters. Lists available scripts and details about them.
`update-fields.sh`	Update a Solr `schema.xml` with a given list of metadata fields. See `update-fields.sh -h` for usage details and example use cases at *Updating the Solr Schema* and *Updating the Solr Schema (Developers)*.
`solr-driver.sh`	Automate updates to a `schema.xml` in a Solr Core. Either run in watch mode or as a oneshot script. See `solr-driver.sh -h` for usage details. Best used as a sidecar or a one-off job.

Solr Template

In addition, at `/template` a Solr Configset is available, ready for Dataverse usage with a tuned core config and schema.

Providing this template to a vanilla Solr image and using solr-precreate with it will create the necessary Solr search index.

The `solrconfig.xml` and `schema.xml` are included from the upstream project `conf/solr/...` folder. You are obviously free to provide such a template in some other way, maybe tuned for your purposes. As a start, the contained script `update-fields.sh` may be used to edit the field definitions.

9.6.4 Build Instructions

Assuming you have Docker, Docker Desktop, Moby or some remote Docker host configured, up and running from here on. Note: You need to use Maven when building this image, as we collate selective files from different places of the upstream repository. (Building with pure Docker Compose does not support this kind of selection.)

By default, when building the application image, it will also create a new config baker image. Simply execute the Maven modules packaging target with activated "container" profile from the projects Git root to build the image:

```
mvn -Pct package
```

If you specifically want to build a config baker image *only*, try

```
mvn -Pct docker:build -Ddocker.filter=dev_bootstrap
```

The build of config baker involves copying Solr configset files. The Solr version used is inherited from Maven, acting as the single source of truth. Also, the tag of the image should correspond the application image, as their usage is intertwined.

Some additional notes, using Maven parameters to change the build and use ...:

- ... a different tag only: add `-Dconf.image.tag=tag`.
  Note: default is `${app.image.tag}`, which defaults to `unstable`
- ... a different image name and tag: add `-Dconf.image=name:tag`.
  Note: default is `gdcc/configbaker:${conf.image.tag}`
- ... a different image registry than Docker Hub: add `-Ddocker.registry=registry.example.org` (see also DMP docs on registries)
- ... a different Solr version: use `-Dsolr.version=x.y.z`

Processor Architecture and Multiarch

This image is published as a "multi-arch image", supporting the most common architectures Dataverse usually runs on: AMD64 (Windows/Linux/...) and ARM64 (Apple M1/M2), by using Maven Docker Plugin's BuildX mode.

Building the image via `mvn -Pct package`, etc. will only build for the architecture of the Docker machine's CPU.

Only `mvn -Pct deploy -Ddocker.platforms=linux/amd64,linux/arm64` will trigger building on all enabled architectures. Yet, to enable building with non-native code on your build machine, you will need to setup a cross-platform builder.

On Linux, you should install qemu-user-static (preferably via your package management) on the host and run `docker run --rm --privileged multiarch/qemu-user-static --reset -p yes` to enable that builder. The Docker plugin will setup everything else for you.

9.6.5 Tunables

This image has no tunable runtime parameters yet.

9.6.6 Locations

Location	Value	Description
${SCRIPT_DIR	/scripts	Place to store the scripts. Part of $PATH.
${SOLR_TEMPL	/template	Place where the Solr Configset resides to create an index core from it.
${BOOTSTRAP_	/scripts/bootstrap	Stores the bootstrapping personas in sub-folders.
${BOOTSTRAP_BASE	/scripts/bootstrap/base	Minimal set of scripts and data from upstream scripts/api folder, just enough for the most basic setup. The idea is that other personas may reuse it within their own init.sh, avoiding (some) code duplication. See dev persona for an example.

9.6.7 Exposed Ports

This image contains no runnable services yet, so no ports exposed.

9.6.8 Entry & Extension Points

The entrypoint of this image is pinned to dumb-init to safeguard signal handling. You may feed any script or executable to it as command.

By using our released images as base image to add your own scripting, personas, Solr configset and so on, simply adapt and alter any aspect you need changed.

9.6.9 Examples

Docker Compose snippet to wait for Dataverse deployment and execute bootstrapping using a custom persona you added by bind mounting (as an alternative to extending the image):

```
bootstrap:
  image: gdcc/configbaker:unstable
  restart: "no"
  command:
    - bootstrap.sh
    - mypersona
  volumes:
    - ./mypersona:/scripts/bootstrap/mypersona
  networks:
    - dataverse
```

Docker Compose snippet to prepare execution of Solr and copy your custom configset you added by bind mounting (instead of an extension). Note that solr-precreate will not overwrite an already existing core! To update the config of an existing core, you need to mount the right volume with the stateful data!

```
solr_initializer:
  container_name: solr_initializer
  image: gdcc/configbaker:unstable
  restart: "no"
  command:
    - sh
```

(continues on next page)

(continued from previous page)

```
    - -c
    - "fix-fs-perms.sh solr && cp -a /template/* /solr-template"
  volumes:
    - ./volumes/solr/data:/var/solr
    - ./volumes/solr/conf:/solr-template
    - /tmp/my-generated-configset:/template

solr:
  container_name: solr
  hostname: solr
  image: solr:${SOLR_VERSION}
  depends_on:
    - dev_solr_initializer
  restart: on-failure
  ports:
    - "8983:8983"
  networks:
    - dataverse
  command:
    - "solr-precreate"
    - "collection1"
    - "/template"
  volumes:
    - ./volumes/solr/data:/var/solr
    - ./volumes/solr/conf:/template
```

CHAPTER

TEN

STYLE GUIDE

This style guide is meant to help developers implement clear and appropriate UI elements consistent with the Dataverse Project's standards.

Contents:

10.1 Foundations

Foundation elements are the very basic building blocks to create a page in Dataverse. Here we will outline how we've applied Bootstrap CSS to our UI, and how the CSS settings in our stylesheet mesh with it. Each section includes links to relevant parts of the official Bootstrap guides and other useful resources, where you can find more detailed documentation. We will also outline other UI resources like FontCustom and Socicon and how they are utilized.

Contents:

- *Grid Layout*
- *Typography*
- *Color Palette*
  - *Brand Colors*
  - *Text Colors*
  - *Link Colors*
  - *Contextual Classes*
- *Icons*
  - *Bootstrap Glyphicons*
  - *Font Custom Icon Font*
  - *Socicon Icon Font*
- *Logos*

10.1.1 Grid Layout

Bootstrap provides a responsive, fluid, 12-column grid system that we use to organize our page layouts.

We use the fixed-width `.container` class which provides responsive widths (i.e. auto, 750px, 970px or 1170px) based on media queries for the page layout, with a series of rows and columns for the content.

The grid layout uses `.col-sm-*` classes for horizontal groups of columns, inside a containing element with a `.row` class. Content should be placed within columns, and only columns may be immediate children of rows.

```
<div class="container">
  <div class="row">
    <div class="col-sm-1">.col-sm-1</div>
    <div class="col-sm-1">.col-sm-1</div>
    <div class="col-sm-1">.col-sm-1</div>
    <div class="col-sm-1">.col-sm-1</div>
    <div class="col-sm-1">.col-sm-1</div>
    <div class="col-sm-1">.col-sm-1</div>
    <div class="col-sm-1">.col-sm-1</div>
    <div class="col-sm-1">.col-sm-1</div>
    <div class="col-sm-1">.col-sm-1</div>
    <div class="col-sm-1">.col-sm-1</div>
    <div class="col-sm-1">.col-sm-1</div>
    <div class="col-sm-1">.col-sm-1</div>
  </div>
  <div class="row">
    <div class="col-sm-8">.col-sm-8</div>
    <div class="col-sm-4">.col-sm-4</div>
  </div>
</div>
```

10.1.2 Typography

The typeface, text size, and line-height are set in the Bootstrap CSS. We use Bootstrap's global default `font-size` of **14px**, with a `line-height` of **1.428**, which is applied to the <body> and all paragraphs.

```
/* bootstrap.css */
body {
  font-family: "Helvetica Neue",Helvetica,Arial,sans-serif;
  font-size: 14px;
  line-height: 1.42857143;
}
```

10.1.3 Color Palette

The default color palette is set in the Bootstrap CSS. It provides the background, border, text and link colors used across the application.

Brand Colors

The Dataverse Project uses a particular color palette to help users quickly and easily identify the different types of objects: Dataverse collections, datasets, and files.

We use our brand color, a custom burnt orange `{color:#C55B28;}`, which is set in our CSS stylesheet, "structure.css". There is also a set of blue "dataset" classes and grey "file" classes, used to help identify those objects when searching and navigating the application.

```
/* structure.css */
.bg-dataverse {
  background:#C55B28;
}
.bg-dataset {
  background:#337AB7;
}
.bg-file {
  background:#F5F5F5;
}

#navbarFixed .navbar-brand {
  color: #C55B28;
}
#navbarFixed .icon-dataverse {
  color: #C55B28;
}
```

```
<div class="bg-dataverse">...</div>
<div class="bg-dataset">...</div>
<div class="bg-file">...</div>
```

```
/* structure.css */
.text-dataverse {
  color:#C55B28;
}
.text-dataset {
  color:#31708F;
}
.text-file {
  color:#F5F5F5;
}
```

```
<p class="text-dataverse">...</p>
<p class="text-dataset">...</p>
<p class="text-file">...</p>
```

Text Colors

Text color is the default setting from Bootstrap CSS.

```
/* bootstrap.css */
body {
  color: #333;
}
```

```
<p>...</p>
```

Link Colors

Link color is the default setting from Bootstrap CSS. The hover state color is set to 15% darker.

Please note, there is a CSS override issue with the link color due to the use of both a Bootstrap stylesheet and a PrimeFaces stylesheet in the UI. We've added CSS such as `.ui-widget-content a {color: #428BCA;}` to our stylesheet to keep the link color consistent.

```
/* bootstrap.css */
a {
  color: #337AB7;
}
a:hover {
  color: #23527C;
}

/* structure.css */
.ui-widget-content a {
  color: #337AB7;
}
.ui-widget-content a:hover, .ui-widget-content a:focus {
  color: #23527C;
}
```

```
<a>...</a>
```

Contextual Classes

Contextual classes from Bootstrap CSS can be used to style background and text colors. Semantic colors include various colors assigned to meaningful contextual values. We convey meaning through color with a handful of emphasis utility classes.

```
<div class="bg-primary">...</div>
<div class="bg-success">...</div>
<div class="bg-info">...</div>
<div class="bg-warning">...</div>
<div class="bg-danger">...</div>
```

```
<p class="text-muted">...</p>
<p class="text-primary">...</p>
```

(continues on next page)

(continued from previous page)

```
<p class="text-success">...</p>
<p class="text-info">...</p>
<p class="text-warning">...</p>
<p class="text-danger">...</p>
```

10.1.4 Icons

We use various icons across the application, which we get from Bootstrap, FontCustom and Socicon. They appear in buttons, in message blocks or as default thumbnails for Dataverse collections, datasets, and files.

Bootstrap Glyphicons

There are over 250 glyphs in font format from the Glyphicon Halflings set provided by Bootstrap. We utilize these mainly as icons inside of buttons and in message blocks.

```
<span class="glyphicon glyphicon-search"></span>
<span class="glyphicon glyphicon-user"></span>
<span class="glyphicon glyphicon-lock"></span>
```

Font Custom Icon Font

With the use of Font Custom we generate our own custom icon fonts. We use these in the search result cards to help distinguish between Dataverse collection, dataset and file results.

```
<span class="icon-dataverse text-dataverse"></span>
<span class="icon-dataset text-dataset"></span>
<span class="icon-file text-file"></span>
```

The *Font Custom* section of the Developer Guide explains how to update these custom icons.

Socicon Icon Font

We use Socicon for our custom social icons. In the footer we use icons for Twitter and Github. In our Share feature, we also use custom social icons to allow users to select from a list of social media channels.

```
<span class="socicon socicon-github" title="Dataverse Project On GitHub"></span>
<span class="socicon socicon-twitter" title="Dataverse Project On Twitter"></span>
<span class="socicon socicon-facebook" title="Dataverse Project On Facebook"></span>
```

10.1.5 Logos

The Dataverse Project logo (below) is displayed in the footer, and was the basis for the creation of the application's icons and favicon.

Create both print and web version of the Dataverse Project logo by downloading this vector-based SVG file: `dataverse_project_logo.svg`

The brand logo (below) was created as a custom icon to represent the concept of a Dataverse collection. It is used as the logo in the Bootstrap navbar component and across the application.

Create both print and web version of the Dataverse collection logo by downloading this vector-based SVG file: `Dataverse_brand_icon.svg`

Here is another vector-based SVG file with three rings: `Dataverse_3ring-brand_icon_EqualSpace.svg`

10.2 Patterns

Patterns are what emerge when using the foundation elements together with basic objects like buttons and alerts, more complex Javascript components from Bootstrap like tooltips and dropdowns, and AJAX components from PrimeFaces like datatables and commandlinks.

Contents:

- *Navbar*
- *Breadcrumbs*
- *Tables*
- *Forms*
- *Buttons*
  - *Action Buttons*
  - *Action Buttons Block*
  - *Form Buttons*
  - *Icon-Only Buttons*
- *Pagination*
- *Labels*
- *Alerts*
  - *Message Classes*
- *Images*
- *Panels*
- *Tabs*
- *Modals*

10.2.1 Navbar

The Navbar component from Bootstrap spans the top of the application and contains the logo/branding, aligned to the left, plus search form and links, aligned to the right.

When logged in, the account name is a dropdown menu, linking the user to account-specific content and the log out link.

```
<nav id="navbarFixed" class="navbar navbar-default navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <a href="#" onclick="return false;">
        <span class="navbar-brand"><i id="icon-dataverse" class="icon-dataverse"></i>␣
↪Dataverse</span>
      </a>
    </div>
    <div class="collapse navbar-collapse" id="topNavBar">
      <ul class="nav navbar-nav navbar-right">
        <li>
          ...
        </li>
      </ul>
    </div>
  </div>
</nav>
```

10.2.2 Breadcrumbs

The breadcrumbs are displayed under the header, and provide a trail of links for users to navigate the hierarchy of containing objects, from file to dataset to Dataverse collection. It utilizes a JSF repeat component to iterate through the breadcrumbs.

```
<div id="breadcrumbNavBlock" class="container" jsf:rendered="#{true}">
  <ui:repeat value="#{page.breadcrumbs}" var="breadcrumb" varStatus="status">
    <h:outputText value=" &gt; " styleClass="breadcrumbCarrot" rendered="#{true}"/>
    <div class="breadcrumbBlock">
      ...
    </div>
  </ui:repeat>
</div>
```

10.2.3 Tables

Most tables use the DataTable components from PrimeFaces and are styled using the Tables component from Bootstrap.

```
<p:dataTable id="itemTable" styleClass="headerless-table margin-top" value="#{page.item}
↪" var="item" widgetVar="itemTable">
  <p:column>
    ...
  </p:column>
</p:dataTable>
```

10.2.4 Forms

Forms fulfill various functions across the site, but we try to style them consistently. We use the `.form-horizontal` layout, which uses `.form-group` to create a grid of rows for the labels and inputs. The consistent style of forms is maintained using the Forms component from Bootstrap. Form elements like the InputText component from PrimeFaces are kept looking clean and consistent across each page.

```
<div class="form-horizontal">
  <div class="form-group">
    <label for="userNameEmail" class="col-sm-3 control-label">
      #{bundle['user.username']}
    </label>
    <div class="col-sm-4">
      <p:inputText id="userName" styleClass="form-control"></p>
    </div>
  </div>
</div>
```

Here are additional form elements that are common across many pages, including required asterisks, icon tooltips, placeholder text, input info message with popover link, and validation error message.

```
<div class="form-group form-col-container col-sm-9 edit-compound-field">
  <div class="form-col-container col-sm-12">
    <p class="help-block">
      <h:outputFormat value="#{bundle.htmlAllowedMsg}" escape="false">
        <f:param value="#{bundle.htmlAllowedTags}"/>
      </h:outputFormat>
    </p>
    <label class="control-label" for="metadata_#{subdsf.datasetFieldType.name}">
      #{subdsf.datasetFieldType.localeTitle}
      <h:outputText styleClass="glyphicon glyphicon-asterisk text-danger" value="" />
      <span class="glyphicon glyphicon-question-sign tooltip-icon" data-toggle="tooltip"␣
↪data-placement="auto right" data-original-title="#{subdsf.datasetFieldType.
↪localeDescription}"></span>
    </label>
    <div>
      <p:inputTextarea value="#{dsfv.valueForEdit}" id="description" tabindex="#{block.
↪index+1}" rows="5" cols="60" styleClass="form-control" />
      <div class="alert-danger" jsf:rendered="#{!empty subdsf.validationMessage}">
          <strong>#{subdsf.validationMessage}</strong>
      </div>
    </div>
  </div>
</div>
```

10.2.5 Buttons

There are various types of buttons for various actions, so we have many components to use, including the CommandButton component and CommandLink component from PrimeFaces, as well as the basic JSF Link component and OutputLink component. Those are styled using the Buttons component, Button Groups component and Buttons Dropdowns component from Bootstrap.

Action Buttons

For action buttons on a page, we include an icon and text label.

```
<div class="btn-group" jsf:rendered="#{true}">
  <button type="button" id="editDataSet" class="btn btn-default dropdown-toggle" data-
↪toggle="dropdown">
    <span class="glyphicon glyphicon-pencil"/> Edit <span class="caret"></span>
  </button>
  <ul class="dropdown-menu text-left">
    <li>
      <h:outputLink> ... </h:outputLink>
    </li>
    <li class="dropdown-submenu pull-left">
      <a tabindex="-1" href="#">Option</a>
      <ul class="dropdown-menu">
        <li>
          <h:link> ... </h:link>
        </li>
        <li>
          <h:link> ... </h:link>
        </li>
      </ul>
    </li>
    ...
  </ul>
</div>
```

Action Buttons Block

For the main actions on a page, we use a container block to group them together. They use the Bootstrap justified button groups style class `.btn-group.btn-group-justified` in order to create a group of buttons that stretch at equal sizes to span the entire width of its parent.

The Bootstrap theme provides a `.btn-primary` style class to highlight the primary action for the user. This stronger color provides extra visual weight and identifies the primary action in a set of buttons on the page. In this example button group from the file page, you can see the Download and Explore options are listed together, providing a more scalable solution to configurable options.

```
<div class="col-xs-4">
  <div id="actionButtonBlock">
    <div class="btn-group btn-group-justified">
      <div class="btn-group">
        <button type="button" id="accessFile" class="btn btn-primary btn-access-file␣
↪dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
```

(continues on next page)

(continued from previous page)

```
            Access File
          </button>
          <ul class="dropdown-menu pull-right text-left">
            <li class="dropdown-header">
              Download Options <span class="glyphicon glyphicon-download-alt"></span>
            </li>
            ...
          </ul>
        </div>
      </div>
      <div class="btn-group btn-group-justified">
        <div class="btn-group">
          <button type="button" id="editFile" class="btn btn-default btn-access btn-edit␣
↪dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
            Edit File <span class="caret"></span>
          </button>
          <ul class="dropdown-menu pull-right text-left">
            <li>
              ...
            </li>
          </ul>
        </div>
      </div>
      <div class="btn-group btn-group-justified">
        <a href="#" onclick="return false;" class="btn btn-default btn-xs btn-contact"␣
↪aria-label="Contact Dataset Owner" title="Contact Dataset Owner">
          Contact Owner
        </a>
        <a href="#" onclick="return false;" class="btn btn-default btn-xs btn-share" aria-
↪label="Share Dataset" title="Share Dataset">
          Share
        </a>
      </div>
    </div>
  </div>
</div>
```

Form Buttons

Form buttons typically appear at the bottom of a form, aligned to the left. They do not have icons, just text labels. The primary button is styled differently.

```
<div class="button-block">
  <p:commandButton id="save" styleClass="btn btn-default" value="#{bundle.saveChanges}"␣
↪action="#{page.save}" update="@form,:messagePanel" />
  <p:commandButton id="cancel" styleClass="btn btn-link" value="#{bundle.cancel}" action=
↪"#{page.cancel}" process="@this" update="@form">
    <p:resetInput target="@form" />
  </p:commandButton>
</div>
```

Icon-Only Buttons

There are a few places where we use icon-only buttons with no text label. For these buttons, we do utilize tooltips that display on hover, containing a text label.

We use the style class `.no-text` with the `.glyphicon` class to fix spacing issues from margins and padding applied to buttons with text labels.

```
<p:commandLink styleClass="btn btn-default btn-sm bootstrap-button-tooltip" title="#
↪{bundle.add}" actionListener="#{Page.add(valCount.index + 1)}">
  <h:outputText styleClass="glyphicon glyphicon-plus no-text"/>
</p:commandLink>
<p:commandLink styleClass="btn btn-default btn-sm bootstrap-button-tooltip" title="#
↪{bundle.delete}" actionListener="#{Page.remove(valCount.index)}">
  <h:outputText styleClass="glyphicon glyphicon-minus no-text"/>
</p:commandLink>
```

Another variation of icon-only buttons uses the `.btn-link` style class from Bootstrap, styling it more like a link while maintaining button behavior. The button group provides space for up to three buttons for a file in the table, and if there are more than three action button, they utilize the "kebab" More Options button dropdown with the `.glyphicon-option-vertical` icon.

```
<div class="btn-group" role="group" aria-label="#{bundle['file.actionsBlock']}">

    <ui:fragment rendered="#{true}">
        <a class="btn-preview btn btn-link bootstrap-button-tooltip" title="#{bundle.
↪preview}"
           href="#{widgetWrapper.wrapURL('/file.xhtml?'.concat(...)}">
           <span class="glyphicon glyphicon-eye-open"/><span class="sr-only">#{bundle.
↪preview}</span>
        </a>
    </ui:fragment>

    <p:commandLink rendered="#{true}" styleClass="btn-download btn btn-link bootstrap-
↪button-tooltip"
                   title="#{bundle.download}"
                   disabled="#{locked ? 'disabled' : ''}"
                   process="@this" update="@widgetVar(popup)" oncomplete="PF('popup').
↪show();">
      <f:actionListener binding="#{pageBean.function()}" />
      <f:actionListener binding="#{pageBean.function()}" />
      <span class="glyphicon glyphicon-download-alt"/><span class="sr-only">#{bundle.
↪download}</span>
    </p:commandLink>

    <div class="btn-group" jsf:rendered="#{true}">
      <a class="btn-explore btn btn-link bootstrap-button-tooltip"
         title="#{bundle.explore}" id="exploreBtn" data-toggle="dropdown" aria-haspopup=
↪"true" aria-expanded="false" data-original-title="#{bundle.explore}">
        <span class="glyphicon glyphicon-equalizer"/><span class="sr-only">#{bundle.
↪explore}</span><span class="caret"></span>
      </a>
      <ul class="dropdown-menu multi-level pull-right text-left" aria-labelledby=
↪"exploreBtn">
```

(continues on next page)

(continued from previous page)

```
        <ui:repeat var="tool" value="#{exploreTools}">
          <li>
            <p:commandLink styleClass="#{locked ? 'disabled' : ''}"
                           disabled="#{locked ? 'disabled' : ''}"
                           action="#{pageBean.function()}">
              #{tool.getDisplayNameLang()}
            </p:commandLink>
          </li>
        </ui:repeat>
      </ul>
    </div>

    <div class="btn-group" jsf:rendered="#{true}">
      <a class="btn-options btn btn-link bootstrap-button-tooltip"
         id="optionsBtn" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false
↪" title="#{bundle.moreOptions}" data-original-title="#{bundle.moreOptions}">
        <span class="glyphicon glyphicon-option-vertical"/><span class="sr-only">#
↪{bundle.moreOptions}</span><span class="caret"></span>
      </a>
      <ul class="dropdown-menu multi-level pull-right text-left" aria-labelledby=
↪"optionsBtn">
        <ui:repeat var="options" value="#{fileOptions}">
          <li>
            <p:commandLink styleClass="#{locked ? 'disabled' : ''}"
                           disabled="#{locked ? 'disabled' : ''}"
                           action="#{pageBean.function()}">
              ...
            </p:commandLink>
          </li>
        </ui:repeat>
      </ul>
    </div>
</div>
```

10.2.6 Pagination

We use the Pagination component from Bootstrap for paging through search results.

```
<ul class="pagination">
  <li class="#{include.page == '1' ? 'disabled' : ''}">
    <h:outputLink value="#{page.page}">
      <h:outputText value="&#171;"/>
      ...
    </h:outputLink>
  </li>
  <li class="#{include.page == '1' ? 'disabled' : ''}">
    <h:outputLink value="#{page.page}">
      <h:outputText value="&lt; #{bundle.previous}"/>
      ...
    </h:outputLink>
  </li>
```

(continues on next page)

(continued from previous page)

```
  ...
  <li class="#{include.page == include.totalPages ? 'disabled' : ''}">
    <h:outputLink value="#{page.page}">
      <h:outputText value="#{bundle.next} &gt;"/>
      ...
    </h:outputLink>
  </li>
  <li class="#{include.page == include.totalPages ? 'disabled' : ''}">
    <h:outputLink value="#{page.page}">
      <h:outputText value="&#187;"/>
      ...
    </h:outputLink>
  </li>
</ul>
```

10.2.7 Labels

The Labels component from Bootstrap is used for publication status (DRAFT, In Review, Unpublished, Deaccessioned), and Dataset version, as well as Tabular Data Tags (Survey, Time Series, Panel, Event, Genomics, Network, Geospatial).

```
<span class="label label-default">Version 2.0</span>
<span class="label label-primary">DRAFT</span>
<span class="label label-success">In Review</span>
<span class="label label-info">Geospatial</span>
<span class="label label-warning">Unpublished</span>
<span class="label label-danger">Deaccessioned</span>
```

10.2.8 Alerts

For our help/information, success, warning, and error message blocks we use a custom built UI component based on the Alerts component from Bootstrap.

```
<div class="alert alert-success" role="alert">...</div>
<div class="alert alert-info" role="alert">...</div>
<div class="alert alert-warning" role="alert">...</div>
<div class="alert alert-danger" role="alert">...</div>
```

Message Classes

Style classes can be added to `p`, `div`, `span` and other elements to add emphasis to inline message blocks.

```
<p class="help-block">
  <span class="text-muted">...</span>
</p>

<p class="help-block">
  <span class="glyphicon glyphicon-ok-sign text-success"></span> <span class="text-
↪success">...</span>
</p>
```

(continues on next page)

(continued from previous page)

```
<p class="help-block">
  <span class="glyphicon glyphicon-asterisk text-info"></span> <span class="text-info">..
↪.</span>
</p>

<p class="help-block">
  <span class="glyphicon glyphicon-warning-sign text-warning"></span> <span class="text-
↪warning">...</span>
</p>

<p class="help-block">
  <span class="glyphicon glyphicon-exclamation-sign text-danger"></span> <span class=
↪"text-danger">...</span>
</p>
```

10.2.9 Images

For images, we use the GraphicImage component from PrimeFaces, or the basic JSF GraphicImage component.

To display images in a responsive way, they are styled with `.img-responsive`, an Images CSS class from Bootstrap.

```
<p:graphicImage styleClass="img-responsive" value="#{Page.imageId}?imageThumb=400" />
```

10.2.10 Panels

The most common of our containers, the Panels component from Bootstrap is used to add a border and padding around sections of content like metadata blocks. Displayed with a header and/or footer, it can also be used with the Collapse plugin from Bootstrap.

```
<div class="panel panel-default">
  <div class="panel-body">
    Basic panel example
  </div>
</div>

<div class="panel panel-default">
  <div data-toggle="collapse" data-target="#panelCollapse0" class="panel-heading">
    <span class="text-info">Panel Heading  <span class="glyphicon glyphicon-chevron-
↪up"/></span>
  </div>
  <div id="panelCollapse0" class="panel-body form-horizontal collapse in">
    <div class="form-group">
      <label class="col-sm-4 control-label">
        Label
      </label>
      <div class="col-sm-6">
        Value
      </div>
    </div>
```

(continues on next page)

(continued from previous page)

```
  </div>
</div>
```

10.2.11 Tabs

Tabs are used to provide content panes on a page that allow the user to view different sections of content without navigating to a different page.

We use the TabView component from PrimeFaces, which is styled using the Tab component from Bootstrap.

```
<p:tabView id="tabView" widgetVar="content" activeIndex="#{Page.selectedTabIndex}">
  <p:ajax event="tabChange" listener="#{Page.tabChanged}" update="@this" />
  <p:tab id="dataTab" title="#{bundle.files}">
      ...
  </p:tab>
  ...
</p:tabView>
```

10.2.12 Modals

Modals are dialog prompts that act as popup overlays, but don't create a new browser window. We use them for confirmation on a delete to make sure the user is aware of the consequences of their actions. We also use them to allow users to execute simple actions on a page without requiring them to navigate to and from a separate page.

Buttons usually provide the UI prompt. A user clicks the button, which then opens a Dialog component or Confirm Dialog component from PrimeFaces that displays the modal with the necessary information and actions to take.

The modal is styled using the Modal component from Bootstrap, for a popup window that prompts a user for information, with overlay and a backdrop, then header, content, and buttons. We can use style classes from Bootstrap for large (`.bs-example-modal-lg`) and small (`.bs-example-modal-sm`) width options.

```
<!-- Large modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target=".bs-
↪example-modal-lg">Large modal</button>

<div class="modal bs-example-modal-lg" tabindex="-1" role="dialog" aria-labelledby=
↪"myLargeModalLabel">
  <div class="modal-dialog modal-lg" role="document">
    <div class="modal-content">
      ...
    </div>
  </div>
</div>
```

10.3 Text

Here we describe the guidelines that help us provide helpful, clear and consistent textual information to users.

Contents:

- *Metadata Text Guidelines*

10.3.1 Metadata Text Guidelines

These guidelines are maintained in a Google Doc as we expect to make frequent changes to them. We welcome comments in the Google Doc.

CHAPTER

ELEVEN

HOW THE GUIDES ARE ORGANIZED

The guides are documentation that explain how to use Dataverse, which are divided into the following sections: User Guide, Installation Guide, Developer Guide, API Guide, Style Guide and Container Guide. The User Guide is further divided into primary activities: finding & using data, adding Datasets, administering dataverses or Datasets, and Dataset exploration/visualizations. Details on all of the above tasks can be found in the Users Guide. The Installation Guide is for people or organizations who want to host their own Dataverse. The Container Guide gives information on how to deploy Dataverse with containers. The Developer Guide contains instructions for people who want to contribute to the Open Source Dataverse project or who want to modify the code to suit their own needs. Finally, the API Guide is for Developers that work on other applications and are interested in connecting with Dataverse through our APIs.

CHAPTER

TWELVE

OTHER RESOURCES

Dataverse Project Site

Additional information about the Dataverse Project itself including presentations, information about upcoming releases, data management and citation, and announcements can be found at https://dataverse.org/

User Group

As the user community grows we encourage people to shares ideas, ask questions, or offer suggestions for improvement. Go to https://groups.google.com/group/dataverse-community to register to our dataverse community group.

Follow Us on Twitter

For up to date news, information and developments, follow our twitter account: https://twitter.com/dataverseorg

Support

We maintain an email based support service that is free of charge. We attempt to respond within one business day to all questions and if it cannot be resolved immediately, we will let you know what to expect.

The support email address is support@dataverse.org.

Reporting Issues and Contributing

Report bugs and add feature requests in GitHub Issues or use GitHub pull requests, if you have some code, scripts or documentation that you'd like to share. If you have a **security issue** to report, please email security@dataverse.org. See also *Reporting Security Issues*.

CHAPTER

THIRTEEN

INDICES AND TABLES

- search

BIBLIOGRAPHY

[preboot] `${CONFIG_DIR}/pre-boot-commands.asadmin`

[postboot] `${CONFIG_DIR}/post-boot-commands.asadmin`

[dump-option] `-XX:+HeapDumpOnOutOfMemoryError`