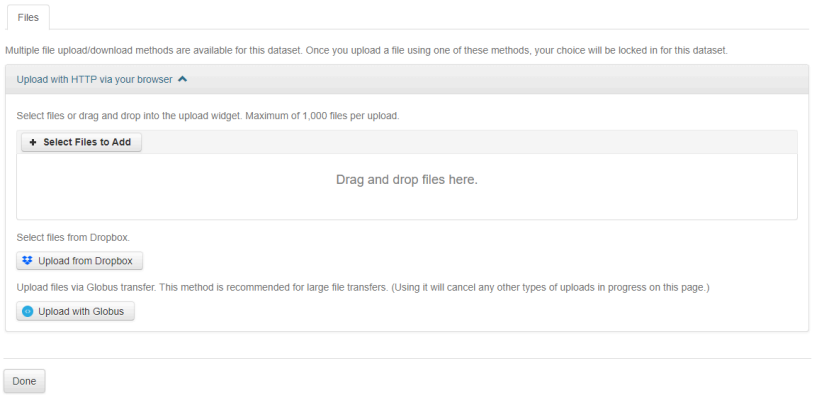
Community Dataverse-Globus Setup and Configuration

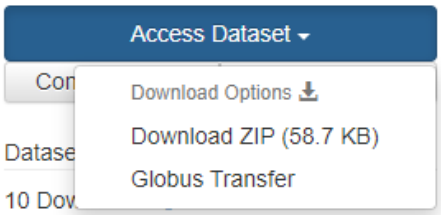
Introduction

Dataverse can be configured with the following components to support large data transfers via Globus:

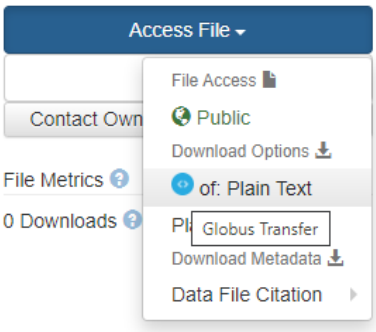
- An object store available as an S3 bucket, configured as a store for one or more collections/datasets within a Dataverse instance
- A Globus Endpoint configured to expose the S3 bucket via FTP/the Globus Service
- The Globus Service providing discovery of other Globus endpoints and coordinating data transfers to/from them
- The Scholars Portal Globus Uploader/Downloader app can be launched from the Dataverse dataset page's Upload panel (shown with the optional Upload from Dropbox

capability also enabled) and from the dataset page ‘Access Dataset’ menu, , respectively:





An entry can also be added to the file download menu via a setting, but this is not yet supported by the released version of the Downloader app:



These applications allow users to select file(s) for transfer to/from other Globus endpoints, initiate those transfers, and, for the upload case, notify Dataverse to add the specified file(s) to the dataset in Dataverse once the transfers are complete.

The overall upload process involves multiple services and, depending on the size of the transfer, can take hours.

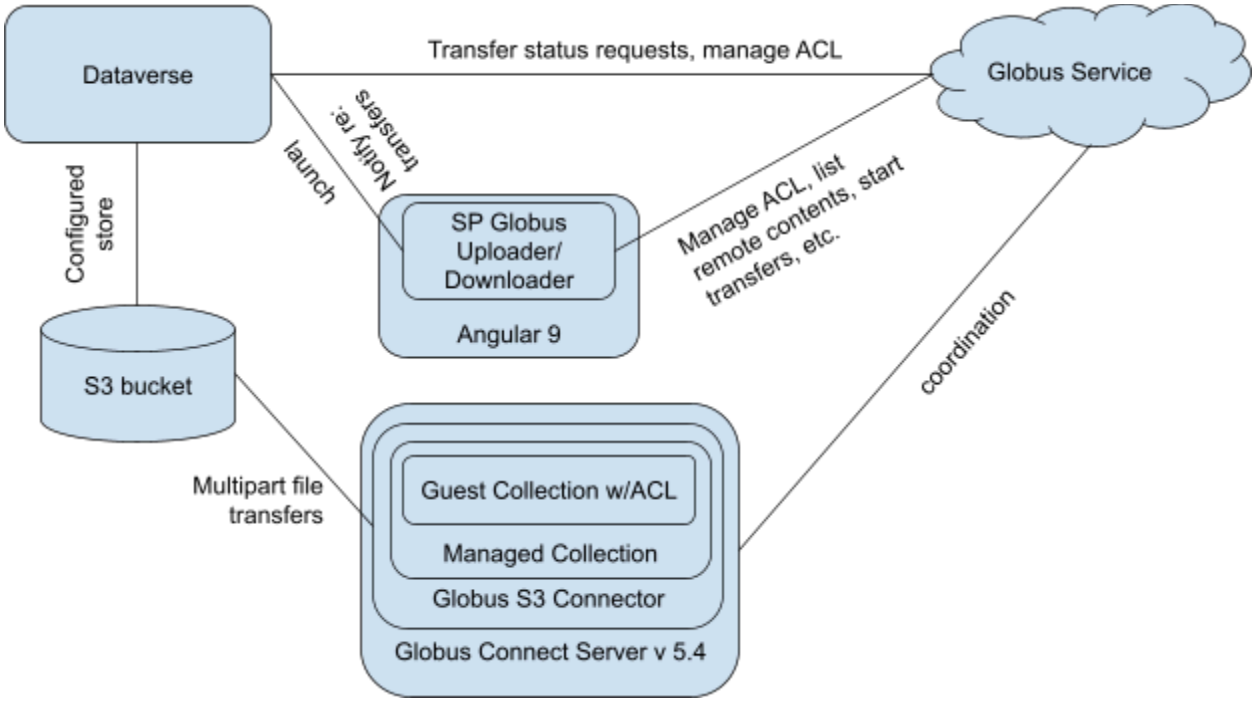


Figure 1: The machines/applications/services involved in using Globus to transfer files to/from Dataverse and their interactions.

Basic Architecture

- Dataverse - this is the standard Dataverse stack including Postgres, Solr, Payara, the Dataverse Java application, etc.
- S3 Bucket - this can be any file/object store available via the S3 protocol. In testing, this can be any bucket, e.g. one at AWS, etc. For production, this would be a large resource, possibly one managed in a research computing organization.
- SP Globus Uploader/Downloader Application. This is an Angular v9 application. For development, this app can be run on the developer's localhost. For production, this needs to be run as a shared application on a machine with a stable https:// address.
- Globus Connect Server (GCS) - this is a Globus-provided application that can be deployed on a standard Linux machine for testing. In production, multiple copies can be set up to support faster (parallel) transfer. The installation scripts and command-line client can be used to configure the additional components on the machine. As part of the process, the Globus service will define a DNS name for the machine and generate a LetsEncrypt certificate for it to support https:// connections.
- Globus Service - this is the https://globus.org service available to individual users and institutions. A globus account is required for setup. This account, and an associated client app account, will be used by Dataverse and the Dataverse Globus app internally. However, data transfers themselves will be authorized via the uploader's own Globus account.

Installation/Configuration:

These instructions assume the existence of a basic Dataverse installation and an existing S3 bucket with associated access/secret keys. They also assume use of an existing globus account for Globus-related setup (called mydataverse@globusid.org in the discussion below.). A familiarity with Globus terminology may also be helpful.

Globus Server, Connector(s), and Collections

Base GCS v 5.4 Server

The development instance of the GCS v5.4 can be run on a "t3a.small" instance in EC2. Whether a larger instance would be useful for MVP production is unclear. (As described here, the primary scaling mechanism to support larger/faster transfers is to add additional transfer nodes, so a small control node is probably a good start. Globus or local Globus administrators may have recommendations.)

The basic installation, along with the prerequisites are described at https://docs.globus.org/globus-connect-server/v5.4/ . Rather than repeat those instructions, this guide highlights details of a few specific steps and indicates additional steps that were required to have a working system during development - you should substitute names/emails/ids appropriate for your instance.

After registering a new GCS server at https://developers.globus.org/ using the mydataverse@globusid.org user, the base server endpoint was setup with:

```
globus-connect-server endpoint setup "GDCC-HDC1-Dev-AWS" \
  --organization "GDCC" \
  --client-id "2791b83e-b989-47c5-a7fa-ce65fd949522" \
  --owner mydataverse@globusid.org \
  --contact-email qqmyers@hotmail.com
```

Response:

```
Created endpoint 2791b83e-b989-47c5-a7fa-ce65fd949522
Endpoint domain_name a08017.7eb0.data.globus.org
No subscription is set on this endpoint, so only basic features are enabled.

To enable subscription features on this endpoint, have the
Globus subscription manager for your organization assign a subscription to this endpoint at
```

https://app.globus.org/file-manager/collections/2791b83e-b989-47c5-a7fa-ce65fd949522/overview/subscription after you've set up at least one node

If you plan on using the Google Drive or Google Cloud Storage
connectors, use
    https://a08017.7eb0.data.globus.org/api/v1/authcallback_google
as the Authorized redirect URI for this endpoint

Note that Globus instructions make it clear that client-ids cannot be reused so the id in the example above must be replaced with a new one from https://developers.globus.org/. The owner should remain mydataverse@globusid.org with the contact email updated to something appropriate. Also note that, as shown in Figure 1, there are multiple named/id components that will be created during the GCS setup. Nominally the top-level Guest Collection will be the only one end users will be interacting with and even then the name will usually be hidden by the Uploader/Downloader tool. However, all of them will be visible to admins and the managed and guest collection names will be publicly visible in Globus (although the only way to get read/write access will be through Dataverse and the Uploader/Downloader app.) For that reason, having names that are globally unique, and branded, and that clearly indicate the level would make sense. (E,g. “Harvard HDC Dataverse Collection” for the top level guest collection and “Harvard HDC GCS” for the GCS endpoint itself would make sense whereas “Example Collection” as shown in the POSIX test section below would be one of MANY in the world.)
Once the server is created, the CLI can then be used to create a transfer node:

```
sudo globus-connect-server node setup \
>     --client-id "2791b83e-b989-47c5-a7fa-ce65fd949522"
```

Response:

```
IP address not specified, using 44.202.202.240
Configuring endpoint
  [########################################]  100%
Starting services
  [########################################]  100%
```

Two steps that were required but not indicated explicitly in the Globus installation guide are:

- Opening the ports required. For ec2 I did this via commands like:
  `aws ec2 authorize-security-group-ingress --group-id sg-<id> --ip-permissions IpProtocol=tcp,FromPort=50000,ToPort=51000,IpRanges='[{CidrIp=0.0.0.0/0}]'`
  and
  `aws ec2 authorize-security-group-ingress --group-id sg-<id> --ip-permissions IpProtocol=tcp,FromPort=443,ToPort=443,IpRanges='[{CidrIp=0.0.0.0/0}]'`
  Note that `--dry-run` can be appended to these commands to test them

- Accept the LetsEncrypt root certificate (Globus automatically creates a DNS name and gets a certificate for the GCS, but in my case machines from which I tried to contact the GCS https:// did not trust the root cert used. I was receiving an error when trying to use the management api (`Error contacting the GCS Manager API SSL error connecting to a08017.7eb0.data.globus.org: CERTIFICATE_VERIFY_FAILEDThis may be because the trust roots for your system don't support the Let's Encrypt CA or another virtual host is configured to use the same server name as the GCS Manager.`) to check the server status. Since Dataverse and the Uploader app talk to the Globus service, I didn't have to do this on all machines, so this is nominally optional/only required to do normal curl calls to the api)
  (thanks to Don Sizemore @ Odum)
  Add https://letsencrypt.org/certs/isrgrootx1.pem to /etc/pki/tls/certs

(Optional Test: Globus POSIX Endpoint)

Although the goal here is to run the S3 Connector, this requires the endpoint to be managed and it may be worth setting up a test collection using the local file storage as a quick test.

This involves setting up the POSIX gateway:
`globus-connect-server storage-gateway create posix "Example Gateway" --domain globusid.org`

Response
`Storage Gateway ID: 7badd4e6-e0af-44db-9a81-d045f44b70e1`

And a collection using that gateway

`globus-connect-server collection create 7badd4e6-e0af-44db-9a81-d045f44b70e1 /home/ "Example Collection"`

Response:
`Collection ID: 6c4461c9-17d3-4899-9c44-bcf51807a6ff`

As configured here, the gateway will only accept users with ids in the globusid.org domain. It may be worth changing this to <your-institution>.edu to support testing with local accounts. This configuration also by default will only allow access to users who have a local unix account matching their id, so, with the mydataverse@globusid.org account, one would also setup a local mydataverse account able to write to their section of the file path shown e.g. /home/hdc1 in this case:

```
[root@ip-172-31-15-74 rocky]# useradd mydataverse
[root@ip-172-31-15-74 rocky]# passwd mydataverse
```

WIth this configured, the mydataverse@globusid.org user should be able to use https://app.globus.org/file-manager , i.e. to login, find this “Example Collection” (one can see the Collection ID in the app to differentiate if there are multiple collections with the same name - unique name preferred), and one with available data (“EuPathDB Public Data" has publicly readable files) and submit a transfer. Upon success the file will be in /home/mydataverse.

Globus S3 Connector

The S3 Connector is considered a “Premium” product only available when your GCS server/endpoint are ‘managed’ and have access to it through an institutional subscription.

Your institution’s subscription may already have access to the S3 Connector. If not, you should contact Globus regarding how to gain access for testing and production. If your institution has access to the S3 connector, you should contact them to make your endpoint ‘managed’, i.e. by sending a support request with the output of the endpoint setup command above and requesting that the endpoint be managed as part.

Once that was approved, I was able to follow the directions at https://docs.globus.org/premium-storage-connectors/v5.4/aws-s3/ to add an S3 storage gateway:

```
globus-connect-server storage-gateway create s3 \
>     "HDC1 Test Storage Gateway" \
>     --domain globusid.org \
>     --s3-endpoint https://s3.amazonaws.com \
>     --s3-user-credential \
>     --bucket gdcc-myers
```

Response:
```
Storage Gateway ID: 04250ac6-a15f-4f6d-bb7c-6e454eaa03f3
```

This example again constrains access to users with globusid.org accounts (like mydataverse) and could be expanded/changed to include *.edu accounts. It points to Amazon’s s3 endpoint and the test gdcc-myers bucket I used. The -s3-user-credential param means that users have to use the globus https://app.globus.org/ to register s3 credentials for their account to use any collections that are created. (In the setup here, only the mydataverse user has to do this - after the next step a guest collection will be created which will allow other users to leverage these credentials).

Globus Managed Collection

With the S3 Connector itself in place, we can now create a specific collection to use:

```
globus-connect-server collection create \
>    04250ac6-a15f-4f6d-bb7c-6e454eaa03f3 \
>    /gdcc-myers/ "HDC1 Test S3 Collection" \
>    --organization 'GDCC' \
>    --contact-email qqmyers@hotmail.com \
>    --description "Initial Data Commons Test" \
>    --allow-guest-collections \
>    --enable-https
```

Response:
Collection ID:
706d285e-dd75-42d0-907c-53020d6713bb04250ac6-a15f-4f6d-bb7c-6e454eaa03f3

The S3 gateway includes the bucket name in the virtual file path so to allow this collection to refer to the bucket root as / the path in the create call has to be /<bucket-name/ as above. (Note that either case can be handled by Dataverse and the Dataverse Globus app with the appropriate settings.)

To use this collection, one has to login to the globus service and add s3 credentials. In this configuration, that should be the mydataverse@globusid.org user. They will need to add the appropriate s3 access and secret keys with the globus service to enable transfers to the bucket. Note that these keys must have the permissions described at - https://docs.globus.org/premium-storage-connectors/v5.4/aws-s3/#permissions-anchor - these are mostly what you'd expect to allow read/write access to the bucket but they also include the ListAllMyBuckets permission (somewhat odd since the configuration here has to specify bucket name(s) explicitly).
This is another step at which testing can be done, i.e. the mydataverse@globusid.org user should be able to transfer files to this new collection and an
`aws s3 ls –recursive <bucketname>`
should show the newly transferred file.

Globus Guest Collection

The final step in setting up the Globus server/service is to create a Guest Collection on top of the Managed Collection created above. Within the Guest collection, anyone given permission, e.g. to read or write in it, or in a virtual sub-folder within it, will leverage the S3 keys stored by the mydataverse@globusid.org user to do so.

The configuration of a guest collection is done through the https://app.globus.org/file-manager. Select the left-column collections tab, select the managed collection created above, click the collections tab on that page and then use the 'Add Guest Collection' button to create the guest collection.

Uploader/Downloader App Installation

Globus Client Registration

Before proceeding to the installation of the app itself - it will need a Globus client id to function. That can be obtained by going to https://developers.globus.org/ while logged in as mydataverse@globusid.org and selecting "Register your app with Globus". In addition to providing a name, the "Native app" option should be selected. This will allow OAuth2 callback URLs to be entered. As a placeholder useful for local testing, you can add http://localhost/upload/* and http://localhost/download/* as a callback URLs. You will also need to generate and store a "secret" - Globus will only show this once.

Give Globus Client Administrator Role

Once you have a client id, that client should be given the Administrator role on the guest collection created above (The app and Dataverse will be using these client credentials to temporarily grant and then revoke write/read permissions to other Globus users as part of the upload-to-Dataverse process. Being an Administrator allows the client to manage those access control lists.) This can be done through https://app.globus.org/file-manager - left-column collections tab, select the guest collection, click the "Roles" tab and click "Assign New Role". On that form the client id, in the form <clientid#>@clients.auth.globus.org can be entered as the Username. (Note that the search may not find this Username, but clicking the "Add" button to the right of that field will enter the client username as written and allow the form to be completed).

Scholars Portal Angular/Globus App:

The SP Globus Uploader/Download App is a stand-alone Angular 9 Javascript/Typescript application that is launched from Dataverse to handle upload or download. Once launched, it allows users to select files from remote Globus endpoints to transfer to Dataverse or vice versa. It generally mirrors the look of Globus's filemanager UI but limits users to initiating a transfer to (or for downloads, from) the specific Dataverse dataset the app was launched from. Under the hood, it manages the access control required to give temporary upload rights on the virtual file path for the given dataset and initiates the file transfer(s). Once those are started the app informs Dataverse and Dataverse then interacts with Globus to monitor the transfers and, upon success, to add those files to the dataset.

The Scholars Portal app is available at https://github.com/scholarsportal/dataverse-globus . The repository/source code should be downloaded/cloned to the desired machine. Make sure you retrieve a version compatible with your version of Dataverse. (For testing, configuring on

localhost (localhost as seen by your web browser) is useful - this is the only name Globus will allow without setting up https/having a certificate.)

Node JS and Angular 9 also have to be installed. On Ubuntu for Windows, I used:

```
dnf module install nodejs:16
npm install -g @angular/cli@9
npm install -g npm@8.8.0

alias ng=/usr/local/bin/ng
```

Once these are installed, cd to the project's root directory and run
`npm install`

As described in the repository README file, the configuration for the app itself is done by editing the ./src/assets/config.json file:
basicGlobusToken - Token of globus application which is base64 encoding of <client ID#>:<client secret> . (Note the client id does not include the '@clients.auth.globus.org' part of the client id. Also note I didn't find a place to get this token and had to do my own base64 encoding.)
globusClientId - ClientId of registered globus application - the same <client ID#> in the basicGlobusToken
globusEndpoint - Globus endpoint (S3 storage) - the endpoint id for the Guest Collection created above
includeBucketInPath - boolean - whether the name of bucket in S3 storage should be added as part of the path used by the endpoint. Depending on how you've set up your managed/guest collections in globus the top-level path accessible may be '/' or /<bucket-name>'. For the latter case, this param should be true.
apiToken - API token of Dataverse superuser. This can be obtained by logging into Dataverse as a superuser (e.g. dataverseAdmin) and creating/copying an API Token It is used for calls to the dataverse API for deleting Globus rules.
redirectUploadURL - set to "http://localhost/upload" , update to https and the real DNS name when configured for production.
The *baseUrl* and *id* can be left blank.

To then run the application, use
`sudo ng serve --port 80`

Dataverse Configuration

To interact with Globus and the Uploader app, Dataverse has to be configured with four new settings.

On the Dataverse machine:

curl -X PUT -d '<same as the basicGlobusToken describe in the section above>' http://localhost:8080/api/admin/settings/:GlobusBasicToken

curl -X PUT -d '<GlobusEndpoint ID for the Guest S3 Collection>' http://localhost:8080/api/admin/settings/:GlobusEndpoint

curl -X PUT -d '<Comma separated list of Globus enabled Dataverse store ids>' http://localhost:8080/api/admin/settings/:GlobusStores

curl -X PUT -d '<Globus App URL>' http://localhost:8080/api/admin/settings/:GlobusAppUrl

Globus also needs to be added to the upload and download methods settings:

curl -X PUT -d 'native/http, globus' http://localhost:8080/api/admin/settings/:UploadMethods

curl -X PUT -d 'globus' http://localhost:8080/api/admin/settings/:DownloadMethods

The interval between Dataverse calls to Globus to check upload status can also be set. The default is 50 seconds.

curl -X PUT -d '300' http://localhost:8080/api/admin/settings/:GlobusPollingInterval

Dataverse S3 Store

For everything here to work, Dataverse must also be configured to use the specified S3 bucket as a 'normal' S3 store. This is well documented in the Dataverse Installation Guide .With this design, files can also be uploaded to this bucket using Dataverse's standard file upload mechanism. The recommended setting would include configuring upload-redirect: true which configures that mechanism to support multipart file uploads which has some parallelism. (This could be useful for intermediate-sized files into the 100GB+ range and would also provide a useful comparison with the Globus transfer mechanism).

Final Test

With everything configured as indicated, one can create a dataset via Dataverse's web interface and invoke the Globus Uploader from the Upload Files panel as shown above. (The operation of the app itself can be seen in this video. Note that the video is out-of-date w.r.t. how the DataverseGlobus app is launched.) One limitation of the Globus transfer mechanism is that Globus transfer is not available during dataset creation - the dataset must be created in Dataverse before one can click Upload Files and use Globus. Once the Globus Uploader is

accessible, a user can select one or more files at a remote endpoint to transfer. Note the user must be logged in with their Globus credential in order to initiate transfers (and to access any non-public files in other endpoints that they may have access to). There are some endpoints with publicly available files (e.g. the EUPathDB). A more rigorous test would be to login as a globus user with access to large files at some local endpoint and verify that they can be transferred via the tool to Dataverse. (As noted above, the App will use its Administrator rights to grant the logged in user temporary write permissions to the Dataverse S3 endpoint. The accessibility of the remote endpoint w.r.t. reading the files is solely dependent on the logged in user's credentials. Also note:

- Since Globus uses OAuth2, you may find that you are already logged in as another user (e.g. mydataverse@globisid.org) if you've been working with Globus in other tabs.
- There are many times when Globus will ask for the app (or its own apps) to be granted permission to take actions for the user. This should be a one-time occurrence with the results cached until revoked by the user.

Production Deployment

For production, the biggest change is to deploy the Uploader app behind an https endpoint with an appropriate DNS name and certificate. As of now, there is no documentation for that part of the installation. Nominally the deployment should be able to leverage common practice for Angular apps and not be Globus or Dataverse Globus app specific. However, if there are questions, the developers at Borealis may be able to provide some guidance.

The Globus Server, Connector, and Collection setup steps described here could also work for production, although more parallel instances of the Globus Server would be desirable. The one change required for a minimal production instance would be to update the redirect URLs for the client using the https://developers.globus.org website to point to the https endpoint assigned to the Dataverse Globus app.