Store and Share using EUDAT B2SHARE REST API

EGI-CF EUDAT training workshop
November 2015

Carl Johan Håkansson
EUDAT Service Area Manager
for Data Access and Re-use

Demo: Sarah Berenji Ardestani
What’s that EUDAT again?

- First EUDAT project 2011-2014
- EU Horizon2020 project 2015-2017
- Building a Collaborative Data Infrastructure
- Data Centres and Research Communities
- Community-driven development
- Service Oriented Architecture
Vision
Supportive services

- AAI with B2ACCESS
- EPIC Persistent Identifiers with B2HANDLE
Why B2SHARE?

Store
Share
Publish
Store in B2SHARE

- Long tail data
- Spreadsheets, images, documents, raw data
- File size currently up to 2GB
- Data that do not fit in with existing data management policies
Share

- Share with colleagues all over the world
- Share research data
- Collaborate
Publish

- Publish research results
- Articles and papers
- Documents
- Raw data
How?

Simplicity by design
B2SHARE today and in the near future

- B2SHARE 1.6.1 in production
- Integrated with B2ACCESS for AAI
- Community metadata schemas
- B2SHARE 2.0 planned for December 2015
  - New Architecture with modular design
  - HTTP API – new version
  - New UI module
  - Invenio 3 backend in collaboration with CERN
  - Prepared for distributed instances
  - Easy installation top priority
  - Integration with B2DROP
Federated AAI and Role Based Access Control

- Federated AAI via EUDAT’s service B2ACCESS
- Self-registration: e-mail address only requirement
- Supporting OpenID (Google, more to come)
- EduGAIN partially supported (fully supported soon)
- Role Based Access Control
- Two access levels: normal users and data managers
- Open for e.g. Citizen Scientists
- Promoting Open Access
APIs

- OAI-PMH for metadata harvesting

- HTTP REST API
  - Scripted interaction
  - Integration with communities (community portals, CMS, other storage systems)
  - Integration with other EUDAT Services
  - Integration with other B2SHARE instances
Remember the vision
B2SHARE 2016

- B2SHARE as Metadata Store with B2SAFE & B2FIND
- Semantic Web Technologies (e.g. annotation)
- EUDAT CDI HTTP REST API
- Graph Database support for metadata?
- Support for other storage systems (object stores)?
- Support for large (massive?) data sets
- Support for information packages (e.g. OAIS SIP, etc)?
Know more?

- Start using: https://b2share.eudat.eu
- Try it: https://trng-b2share.eudat.eu
- Read more: http://eudat.eu
How to register and get an access token

- Click "Login" (DO NOT USE "REGISTER NOW")
- Click "Sign in with B2ACCESS" (Do not use login form)
- Now your are in B2ACCESS – use federated login or create id in B2ACCESS for password login
- When back in B2SHARE - click on your user name
- Click "Account", then choose "Application"
- Under "Personal access token", click "New token"
- Name it and create it
- Copy this token to use it
Click "Login"
Click "Sign in with B2ACCESS"
Back in B2SHARE - click your user name
Click "Account"
Choose “Application” in left menu
Under “Personal access token”, click “New token”
Name and create token

New personal access token

Name:
Name of personal access token.

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

[Create]
Name and create your token, copy it to use it.
API Requests: list records

List all the records
List all the records, without any filtering.

- HTTP method: GET
- URL path: /api/records
- Required parameters: access_token
- Optional parameters: page_size, page_offset

Example: curl -i http://example.org/api/records?access_token=LKR35GP7TF&page_size=5&page_offset=2

List records per community
List all records of a specific community.

- URL path: /api/records/COMMUNITY_NAME
- HTTP method: GET
- Required parameters: access_token
- Optional parameters: page_size, page_offset
- Returns: the list of records (in JSON format) or an error message with the list of valid community identifiers if the COMMUNITY_ID is invalid.

Example: curl -i http://example.org/api/records/BBMRI?access_token=LKR35GP7TF&page_size=10&page_offset=3
API: list a specific record

List a specific record

List the metadata of the record specified by RECORD_ID

- URL path: /api/record/RECORD_ID
- HTTP method: GET
- Required parameters: access_token

Example: curl -i http://example.org/api/record/1?access_token=LKR35GP7TF
API: create a record and add a file

Create a new deposition

Create a new deposition

- URL path: /api/depositions
- HTTP method: POST
- Required parameters: access_token
- Returns: the URL of the deposition (both as JSON and in the field `Location` in the http header)

Example: curl -i -X POST http://example.org/api/depositions?access_token=LKR35GP7TF

Upload a new file into a deposition object

Upload a new file into a deposition object

- URL path: /api/deposition/DEPOSITION_ID/files
- HTTP Method: POST
- Required input data: the file, sent as multipart/form-data
- Required parameters: access_token
- Returns: the name and size of the newly uploaded file

Example: curl -i -X POST -F file=@TestFileToBeUploaded.txt http://example.org/api/deposition/23k85hjfluzy346/files?access_token=LKR35GP7TF
API: list the files and commit the record

List the files uploaded into a deposition object

List the files uploaded into a deposition object
- URL path: /api/deposition/DEPOSITION_ID/files
- HTTP Method: GET
- Required parameters: access_token
- Returns: the name and size of all the files in the deposition object

Example: curl -i http://example.org/api/deposition/23k8jfljuy345/files?access_token=LKR35GP7TF

Commit deposition

This action transforms a deposition into an immutable record.
- URL path: /api/deposition/DEPOSITION_ID/commit
- HTTP Method: POST
- Required input data: the metadata for the record object to be created. This metadata must be sent as a list of fields (key: value pairs). The required fields are:
  - domain [string]: the id of the community to which the record belongs
  - open_access [boolean]: the access restriction of the new record
  - title [string]: the title of the new record
  - description [string]: the description of the new record

Depending on the domain specification, other fields could be required in order to make a successful commit. The list of all the fields, with their description, multiplicity and controlled vocabulary, is automatically returned to the user in case one of the required fields is missing.
- Required parameters: access_token
- Returns: the location URL of the new record if the submitted metadata is valid; otherwise, the list of all the metadata fields that can be filled in and details on each one.

Example: curl -i -X POST -H "Content-Type: application/json" -d '{"domain": "generic", "title": "REST Test Title", "description": "REST Test Description", "open_access": "true"}
http://.../api/deposition/DEPOSITION_ID/commit/?access_token=LKR35GP7TF