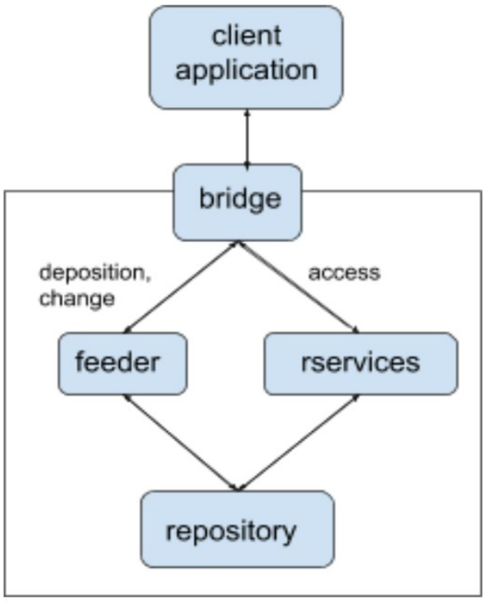
docuteam bridge: documentation for client applications

docuteam bridge v1.0, 12.03.2019

goal

Clients – mostly client applications, but also individuals – should be able to submit a deposition (data and metadata) to our ingest plattform. Depositions will be picked up by docuteam feeder workflows, usually processing and eventually storing the information in a repository. After a successful ingest, feeder (and subsequently bridge) return PIDs for every object (file, folder) within the deposition. Using these PIDs, the client will be able to access (read and change) the deposited objects.



key points

- Bridge is a set of rest APIs that respond in JSON (and binary data)
- Bridge is agnostic to package format
  - Use the simple, bagit-based format docuteam dublin core 1.0 (see Appendix A)
  - Use Matterhorn METS (see Specification)
  - Use other formats like e. g. eCH-0160, SEDA
- Bridge is composed of 3 APIs:
  - ***depositions***: deposition of new packages
  - ***access***: read data deposited with success in the repository
  - ***changes***: update or purge objects in the repository

introduction to the apis

For security reasons, bridge uses only https. Access is restricted via tokens that must be transmitted with each request via the “token” parameter (regardless the HTTP method: get, post, put, patch, delete). Roles are associated to tokens and limit their scope of operation.

- *Depositions* use internal IDs
  - upon success, the PIDs, which are the repository’s persistent IDs and different from bridge internal IDs, are made available via the feeder response field for the whole tree of the deposited objects
- ***Changes*** must imperatively target (for update or purge) one single object in the repository identified by a PID. They can be limited to:
  - metadata (data is unchanged, only for updates)
  - data (metadata is unchanged)
  - object (metadata+data)
- ***Access***
  - is a proxy to docuteam rservices

0 - authentication

- A authentication token must be at least 15 characters long.
- Tokens are passed as the “token” HTTP parameter, for example when using GET, this results in urls of the form: `http://server/access/sync_original/:pid?token=123456789012345`

roles

There are 5 roles:
- The 3 first roles are limited to the organization they are bound to:
  - read (is limited to access api)
  - create (same as read, can also list and create depositions via the deposition API)
  - manage (same as create, can in addition update or delete repository objects via the nodes API)
- The 2 last roles are not limited to any organization
  - admin (authentication i.e. token administration via the GUI)
  - feeder (super user, can do anything, including all deposition status updates)

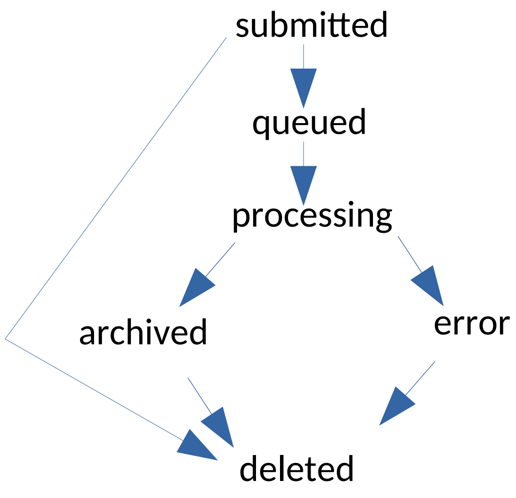
Recommendation:
- use only one token with read roles in applications that do not create nor update depositions,
- use only one token with create role in application that create depositions but do not use the changes api,
- use only one token with manage role in applications that use the changes api.

1 - depositions API

deposition statuses

- *submitted* (deposition received in bridge)
- *queued* (deposition has been attributed to a queue by feeder)
- *processing* (deposition has been downloaded by feeder, that is processing it, i.e. the status is not yet updated to archived)
- *archived* (deposition successfully processed, e.g. stored in the repository, binary object purged)
- *error* (something went wrong, see “message” fields in response)
- *deleted* (deleted from bridge, not from the repository)

Deposition status can be managed by the role “feeder”. Other non-reader roles may only delete depositions (set the status to deleted), except when status is set to “processing”.



Responses

- Responses are expressed in json or a binary format. The json responses contain a list of depositions metadata. Generic structure:

```
{“api”:
  { “name”: “docuteam bridge”,
    “version”: 1.0.0 },
 “response”:
  [{“id”: “id”,
    “uploaded_at”: “2018-11-03T11:13:39.278026Z”,
    “queued_at”: “2018-11-03T14:16:12.678056Z”,
    “processed_by_feeder_at”: “2018-11-03T14:16:12.678016Z”,
    “archived_at”: “2018-11-03T14:16:12.678016Z”,
    “deleted_at”: null,
    “status”: “archived”,
    “feeder_response”: { json-blackbox },
    “organization”: “museumplus”,
    “repository_key”: “museumplus”,
    “package_format” : “DocuteamDublinCore1.0”,
    “package_attached” : true,
    “package_byte_size”: 2716786
   }]
  “request”:
    {“organization”: “museumplus”,
     “role”: “reader”,
     “requested_at”: “2018-11-03T11:13:39.278026Z”}
}
```

NB:

- “id” is the deposition id
- the “feeder_response” is also in json format, but a black box for bridge, which only makes sure it is valid json
- suggested structure for the feeder response:

```
{‘pids’:[
   { "clientId":"id1", "pid":"pid1"},
   { "clientId":"id2", "pid":"pid2"},
    ... ],
‘message’: ’Error processing clientId1: Adobe DRM is not  supported’
‘feeder_version’: ‘5.0’}
```

Overview (routes):

```
depositions GET      /depositions          depositions#index
            POST     /depositions          depositions#create
            GET      /depositions/:id      depositions#show
            PATCH    /depositions/:id      depositions#update
            PUT      /depositions/:id      depositions#update
```

Details:

- Create
  - POST   /depositions   depositions#create
  - Optional params:
    - package_format, it is set by default to “MatterhornMets”, for Museum+ use “DocuteamDublincore1.0”
  - Constrains
    - Mandatory: the package binary data must be sent with request
  - Response: json (see above).
  - Example with curl to deposit the sip.zip package (sip.zip is the filename):
    - curl -X POST -F "package=@sip.zip" “https://bridge-stage.docuteam.ch/depositions?token=123456789012345t&package_format=DocuteamDublincore1.0”
- Index/List (only allowed for the token that has create the deposition)

- GET /depositions depositions#index
  (lists all depositions of user)
  - GET example /depositions?status=<status>&from=<datetime>&to=<datetime>&user=<token>
  - Optional params:
    - id (deposition id)
    - status,
    - from, until, (format: YYYY-MM-DD)
    - user (via user token). Note: Non-admin users are automatically restricted to their user's depositions.
    - organization (will return all depositions made by any token linked to specified organisation). Note: Non-admin users are automatically restricted to their user's depositions hence to their own organisation.
  - Examples with curl:
    - curl "https://bridge-stage.docuteam.ch/depositions?token=123456789012345t"
    - curl "https://bridge-stage.docuteam.ch/depositions?token=123456789012345t&status=submitted"
    - curl "https://bridge-stage.docuteam.ch/depositions?token=123456789012345t&status=submitted&organization=customerx"
    - curl "https://bridge-stage.docuteam.ch/depositions?token=123456789012345t&from=2018-11-01&until=2018-11-30"
    - curl "https://bridge-stage.docuteam.ch/depositions?token=123456789012345t&status=error&from=2018-11-01"
    - curl "https://bridge-stage.docuteam.ch/depositions?token=123456789012345t&id=54321"
- Get package binary data back from deposition (warning: by opposition to the repository bridge is a temporary storage and this binary data is delete upon successful archiving of deposition)
  - GET /depositions/:id depositions#show
    - Delivers binary data
  - Example with curl (to get back package and write it in sip.zip)
    - curl "https://bridge-stage.docuteam.ch/depositions/1?token=123456789012345t" --output sip.zip
- Update deposition (this operation is restricted and mainly called by feeder)
  - PUT /depositions/:id depositions#update
    - Update (only allowed for the token that has created the deposition), enables users to delete a deposition (N.B. binary data is purged from bridge upon delete)
    - Params
      - id (the deposition id to be update)
      - status=deleted
    - Note: the feeder super role has access to additional update actions When feeder sets the status to archived the package binary data is deleted from bridge.
      - status=[deleted|queued|processing|archived|error] Data
      - feeder_response (json string, see structure above), this parameter is mandatory to set the status to "archived" or

        "error".
- Example with curl to update the status of a submission
    - curl -X PUT "https://bridge-stage.docuteam.ch/depositions/12345?token=123456789012345t&status=deleted"
    - curl -X PUT "https://bridge-stage.docuteam.ch/depositions/12345?token=1234567890feeder&status=queued"
    - curl -X PUT "https://bridge-stage.docuteam.ch/depositions/12345?token=1234567890feeder&status=processing"
    - curl -X PUT "https://bridge-stage.docuteam.ch/depositions/12345?token=1234567890feeder&status=archived&feeder_response=%7B%22pids%22%3A%5B%221%22%2C%222%22%5D%7D"
    - curl -X PUT "https://bridge-stage.docuteam.ch/depositions/12345?token=1234567890feeder&status=error&feeder_response=%7B%22message%22%3A%22The+Error%22%7D"

2 - access API

This API is a of proxy to docuteam rservices.

API specification

Synchronous access;

access/sync_preview
GET /access/sync_preview/:pid   sync_preview#download

Example with curl:

curl
“https://bridge-stage.docuteam.ch/access/sync_preview/
test:38?token=123456789012345r”

access/sync_original
GET /access/sync_original/:pid   sync_original#download

Example with curl:

curl
“https://bridge-stage.docuteam.ch/access/sync_preview/
test:38?token=123456789012345r” --output original.pdf

access/sync_dip
GET /access/sync_dip/:pid   sync_dip#download

`Optional parameters for sync_dip:`

- recursively=(true|false) : will return a package with all children objects, by default it is set to false
- verifyChecksum=(true|false) : will verify the checksums of all returned objects, by default it is set to false. When activated, this may considerably slow down requests.

Example with curl:

- curl “https://bridge-stage.docuteam.ch:3000/access/sync_dip/test:38?token=123456789012345r” --output dip.zip
- curl “https://bridge-stage.docuteam.ch:3000/access/sync_dip/test:38?recursively=true” --output dip.zip

access/sync_metadata
GET /access/sync_metadata/:pid   sync_metadata#download

Example:

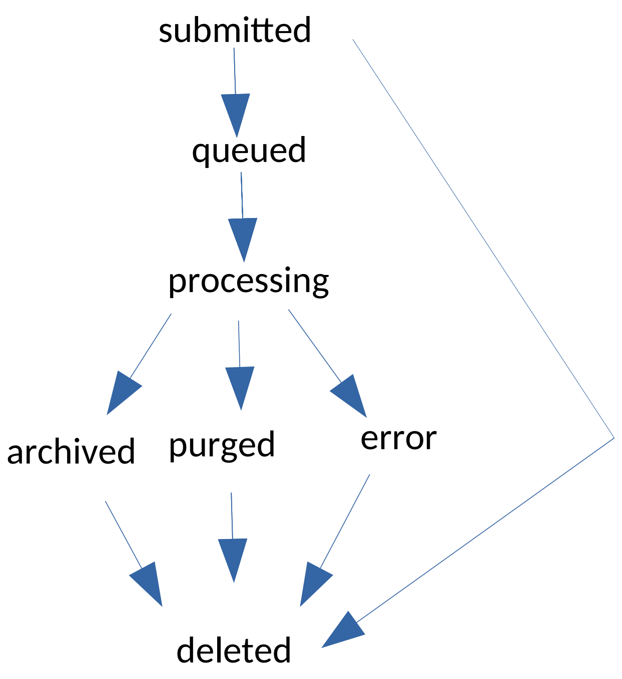
- curl
“https://bridge-stage.docuteam.ch:3000/access/sync_metadata/test:38?token=123456789012345r”
- curl
“https://bridge-stage.docuteam.ch:3000/access/sync_metadata/test:38?recursively=true”

3 - changes API

Updates or deletes objects in the repository.

Changes statuses

- *submitted* (new change corresponding to an update or purge was created)
- *queued* (change has been attributed to a queue by feeder)
- *processing* (change has been downloaded by feeder, that is processing it, i.e. the status is not yet updated or purged)
- *archived* (change successfully processed, e.g. the object stored in the repository was updated)
- *purged (object* successfully purged from the repository)
- *error* (something went wrong, see “message” fields in response)
- *deleted* (change deleted from bridge, only admins can access deleted changes)



Overview

The Changes json responses are similar to the Deposition responses, there are two additional fields:

- “pid” that relates to the repository targeted id in the archive
- “task” that describes the action

Responses

- Responses in json or binary format. Generic structure for json responses:

```
{“api”:
  { “name”: “docuteam bridge”,
     “version”: 1.0.0 },
  “response”:
    [{“id”: “id”,
     “uploaded_at”: “2018-11-03T11:13:39.278026Z”,
     “queued_at”: “2018-11-03T14:16:12.678560Z”,
     “processed_by_feeder_at”: “2018-11-03T14:16:12.678016Z”,
     “archived_at”: “2018-11-03T14:16:12.678016Z”,
     “purged_at”: null,
     “deleted_at”: null,
     “status”: “archived”,
     “feeder_response”: { json-blackbox },
     “organization”: “museumplus”,
     “repository_key”: “museumplus”,
     “package_format” : “DocuteamDublinCore1.0”,
     “package_attached” : true,
     “package_byte_size”: 2716786,
     “task” : “node_update”,
     “pid” : “ns:87654”
    }]
   “request”:
     {“organization”: “museumplus”,
      “role”: “reader”,
      “requested_at”: “2018-11-03T11:13:39.278026Z”}
  }
```

details

- Create a request for change
  - POST /changes/:id(.:format) changes#update
  - here, the id refers to the repository PID
  - Optional params:
    - package_format, it is set by default to “MatterhornMets”, for Museum+ use “DocuteamDublincore1.0”
  - Mandatory params:
    - task (data_update | metadata_update | object_update)
    - or task (data_delete | object_delete), Note: object_delete (data and metadata will be deleted.
- Index/List (only allowed for the token that has create the deposition)
  - GET /changes changes#index

    (lists all depositions of user)
    - GET example /changes?status=<status>&from=<datetime>&to=<datetime>&user=<token>
    - Optional params:
      - id (deposition id)
      - status,
      - from, until, (format: YYYY-MM-DD)
      - user (via user token). Note: Non-admin users are automatically restricted to their user’s depositions.

        - organization (will return all changes made by any token linked to specified organisation). Note: Non-admin users are automatically restricted to their user’s changes hence to their own organisation.
    - GET   /changes/:id changes#show
        - Delivers binary data (or 404 and warning if not present)
- Update (only allowed for the token that has created the change), enables users to delete a change (notably, binary data is purged from bridge)
    - PUT /changes/:id depositions#update
    - Params
        - Id (the deposition id to be update)
        - status=deleted
    - Note: the feeder super role has access to additional update actions. When feeder sets the status to archived the package binary data is deleted from bridge.
        - status=[deleted|queued|processing|archived|purged|error] Data
        - feeder_response (json string, see structure above), this parameter is mandatory to set the status to “purged”, “archived” or “error”.

Appendix A

package structure: docuteam dublin core1.0

container format

- A zipped bagit, with at least the sha256 checksums (other checksum algorithms supported by bagit are optional)
- Bagit library:
    - https://tools.ietf.org/id/draft-kunze-bagit-14.txt
    - https://github.com/LibraryOfCongress/bagit-spec
    - https://github.com/LibraryOfCongress/bagit-python
    - https://github.com/LibraryOfCongress/bagit-java

- Container metadata is expressed in XML DublinCore
    - http://dublincore.org/documents/dcmi-terms/

container structure (inside the zipped bagit)

1. the root folder, corresponding to the "object", must be named "data"
2. subfolders may be named freely
3. subfolders may be organized recursively
4. in each folder (at all levels) there is a mandatory metadata file always named "dc.xml"
5. in addition, each folder (at all levels) may contain either (but not both!):
    a. one or more subfolders
    b. one datafile, which may be named freely (except "dc.xml")

A somewhat more formal structure definition :

```
<rootfolder>        ::= <metadata file> <children>*
<folder>            ::= <metadata file> <children>*
<children>          ::= <folder>* | <file>
<metadata file>     ::= dc.xml
<file>              ::= filename.ext
```

example1 : container structure with only one file

- root-folder:"data"
    - file:dc.xml
    - file:name1.extension

example2 : container structure with several files

- root-folder:"data"
    - file:dc.xml
    - sub-folder:name1
        - file:dc.xml
        - file:name1.extension
    - sub-folder:name2
        - file:dc.xml

        - file:name2.extension
    - sub-folder:name3
        - file:dc.xml
        - file:name3.extension

example3 :complex structure with several files

- root-folder:"data"
    - file:dc.xml
    - sub-folder:name1
        - file:dc.xml
        - sub-sub-folder:name2
            - file:dc.xml
            - sub-sub-sub-folder:name3
                - file:dc.xml
                - file:name3.extension
        - sub-sub-folder:name4
            - file:dc.xml
            - sub-sub-sub-folder:name5
                - file:dc.xml
                - file:name5.extension
    - sub-folder:name6
        - file:dc.xml
        - file:name6.extension
    - sub-folder:name7
        - file:dc.xml
        - sub-sub-folder:name8
            - file:dc.xml
            - sub-sub-sub-folder:name9
                - file:dc.xml
                - file:name9.extension

metadata constraints

This version 1.0 of the package format is restricted to the Dublin Core Metadata Element Set, limited to 15 elements (dc 1.1 terms, see http://dublincore.org/documents/dcmi-terms/#section-3). In addition, the following constraints apply:

1. The **"Identifier" field is mandatory at each level in "dc.xml", it must contain:**
    - **At each level: the the client application identifier** of the object with the prefix "clientid:" e.g. "clientid:1234567" or "clientid:d4FTw3v6T"
    - **At root level, a mandatory identifier** with the customer namespace in the repository (this is often the ISIL code) prefixed with "namespace:", e.g. "namespace:CH-1234-1"
2. The **"Title" field is mandatory at each level in the "dc.xml" file**. It is not repeatable.
3. **All other 13 fields are optional and repeatable**, they are:
    - Creator (e.g. the authors, one per field repetition, that can be persons or institutions)
    - Subject (typically keywords, one per field repetition)
    - Description (a textual description of the object or folder)
    - Publisher
    - Contributor
    - Date (use ISO8601, e.g. 2018-11-30)

- Type
- Format
- Source
- Language
- Relation
- Coverage
- Rights