docuteam bridge

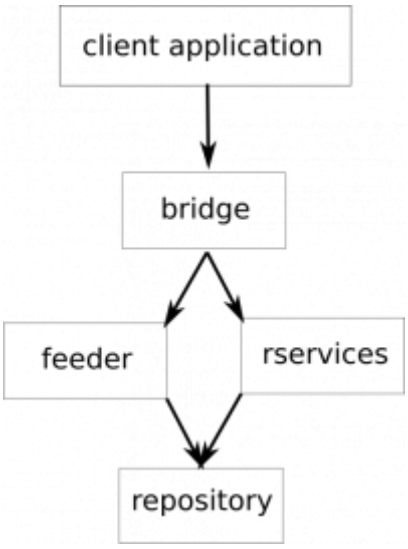
Documentation for docuteam bridge v1.0.0

Goal

Client applications should be able to submit a deposition (a package with data and metadata) to our ingest platform. Depositions are then picked up by docuteam feeder workflows, usually processing and storing the deposition in a preservation repository. After a successful ingest, feeder (and subsequently bridge) return persistent identifiers (PIDs) for every object (file, folder) within the deposition. Using these PIDs, clients are able to access the deposited objects again.

In the Open Archival Information System (OAIS) terminology:

- docuteam bridge receives Submission Information Packages (SIP) on its depositions API.
- The SIPs are processed by docuteam feeder (quality assurance, optional initial preservation actions) and stored into a repository as Archival Information Packages (AIP).
- Client applications retrieve Dissemination Information Packages (DIP) of their originally submitted objects using the access API of docuteam bridge.



Overview

- bridge is a set of REST APIs that usually return a JSON (and binary data) response.
- bridge is agnostic to the package format of the deposition, e.g.:
  - Use the simple, BagIt-based format docuteam dublin core (see SIP docuteam dublin core)
  - Use Matterhorn METS (see its specification or the registered METS profile)
  - Use other formats like eCH-0160 or SEDA.
- bridge consists of three APIs:
  - **depositions**: deposition of new packages
    - New depositions are temporarily stored by bridge. They are made available to feeder, which in turn processes and stores them into the repository.

        - Depositions are identified by bridge IDs. These identifiers are returned with the HTTP response of a new deposition request.
        - Upon successful archiving in the repository,
            - the status of the deposition is set to „archived“,
            - persistent identifiers (PIDs) for each of the deposition's objects are returned by bridge,
            - and the SIP is deleted from the depositions as it is now preserved in the repository.
    - **changes** (planned): update or purge specific repository objects
        - metadata only (data remains unchanged)
        - data only (metadata remains unchanged)
        - object (metadata and data are modified)
    - **access**: read/access data successfully deposited into the repository
        - Gives access to the full DIP corresponding to a deposited SIP.
        - Gives access to metadata, data or previews of specific repository objects.

Authentication

Access is restricted via tokens that must be transmitted with each request using the „token“ parameter. This is required for all HTTP methods, i.e. GET, POST, PUT, DELETE. Bridge relies solely on tokens for authentication an authorization. Tokens are bound to organizations and roles and restrict the API.

- An authentication token must be at least 15 characters long.
- Tokens are passed using a „token“ HTTP request parameter, i.e. `https://server:port/api/method?token=123456789012345`

Roles

Roles are associated to tokens and limit their scope of operation.

There are five roles:

- The following three roles are limited to a single organization:
    - **read**: is restricted to the access API
    - **create**: has the same authorizations as read, but can in addition list, create and delete depositions via the deposition API
    - **manage**: has the same authorizations as create, and can in addition update or purge repository objects via the change API
- The following two roles have a global scope:
    - **admin**: authentication and authorization management, i.e. token administration using the API or the GUI
    - **feeder**: super user, can do anything except token administration, including status updates for depositions of any organization.

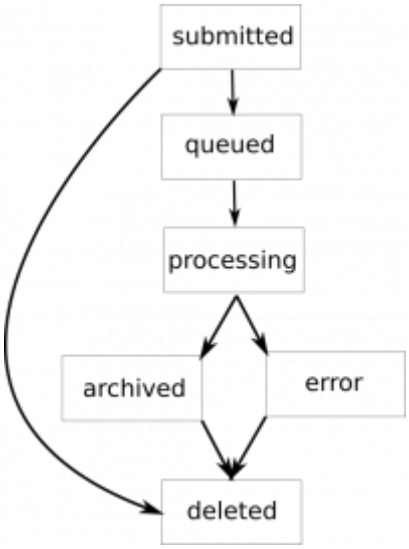
Depositions API

Status Model

Depositions have one of the following status:

- **submitted**: the deposition was received by bridge
- **queued**: the deposition has been queued for processing by feeder
- **processing**: the deposition has been downloaded by feeder, which is processing it (validation, quality assurance, preservation actions and storage into the repository)
- **archived**: the deposition was successfully processed by feeder and archived into the repository. This implies that:
  - Persistent identifiers (PIDs) allocated to the deposition's objects are available in bridge. Each digital object in the repository (folder, file) gets its own PID, and relations between the customer's application IDs (if contained in the original deposition) and PIDs are explicit in the deposition metadata.
    PIDs are required to access (read, modify, purge) repository objects through the access API.
  - The originally deposited package has been deleted from bridge.
- **error**: something went wrong during the deposition's processing by feeder. In this case, the „message“ fields of the deposition will contain error information.
- **deleted**: the deposition was deleted from bridge.
  This only means that the deposition was removed from bridge, not from the repository.

The deposition status can only be managed directly by the role „feeder“. Other non-reader may only set the status implicitly by creating or deleting depositions. Deleting depositions (set the status to deleted) is blocked when status is either „queued“ or „processing“.



Routes

```
POST   /depositions       depositions#create
GET    /depositions       depositions#index
GET    /depositions/:id   depositions#show
```

```
PUT    /depositions/:id   depositions#update
```

Create

Creates a new deposition.

Allowed calls

POST /depositions

Requirements

Token with role `create`, `manage`, or `feeder`.

Parameters

- `token`
- `package_format` ⇒ indicate the format of the submitted package, e.g. „MatterhornMets“ or „DocuteamDublincore1.0“ (optional, default: „MatterhornMets“)
- `[binary data]`

Examples

```
curl -X POST -F "package=@sip.zip"
"https://bridge.docuteam.ch/depositions?token=123456789012345&package_format
=DocuteamDublincore1.0"
```

Index

Lists/shows the existing depositions with details.

Allowed calls

GET /depositions

Requirements

Token with role `create`, `manage`, or `feeder`.

Parameters

- `token`
- `id` (optional)
- `status` (optional)
- `from` (optional, format:YYYY-MM-DD)
- `until` (optional, format:YYYY-MM-DD)
- `organization` (optional)

Examples

```
curl "https://bridge.docuteam.ch/depositions?token=123456789012345"
curl "https://bridge.docuteam.ch/depositions?token=123456789012345&id=2"
curl
"https://bridge.docuteam.ch/depositions?token=123456789012345&status=submitt
ed"
curl
"https://bridge.docuteam.ch/depositions?token=123456789012345&status=submitt
ed&organization=XY"
curl
"https://bridge.docuteam.ch/depositions?token=123456789012345&from=2018-11-0
1&until=2018-11-30"
curl
"https://bridge.docuteam.ch/depositions?token=123456789012345&status=error&f
rom=2018-11-01"
```

Show

Retrieve the binary content of a deposition.

Allowed calls

GET /depositions/:id

Requirements

Token with role `create`, `manage`, or `feeder`.

Parameters

- `:id` deposition ID
- `token` (mandatory)

Examples

```
curl "https://bridge.docuteam.ch/depositions/1?token=123456789012345" --
output sip.zip
```

Update

Set status and processing details. By setting the status to `deleted`, the depositions binary content will be removed.

Allowed calls

PUT /depositions/:id

Requirements

Token with role `create`, `manage` (limited to delete a deposition), or `feeder`.

Parameters

- `:id` deposition ID
- `token`
- `status` ⇒ supported values are `deleted | queued | processing | archived | error`
- `feeder_response` (optional, url encoded string)

Examples

```
curl -X PUT
"https://bridge.docuteam.ch/depositions/12345?token=123456789012345&status=d
eleted"
curl -X PUT
"https://bridge.docuteam.ch/depositions/23?token=12super34token56&status=arc
hived&feeder_response=%7B%22pids%22%3A%5B%221%22%2C%222%22%5D%7D"
```

Responses

Responses are given in JSON or (for the show method) as a binary. Each JSON response is a list of deposition with their details. The generic structure looks like this:

```
{ "api" :
   { "name": "docuteam bridge",
```

```
     "version": "v1.0.0" },
 "response" :
   [
     { "id": 1234,
       "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"}
}
```

Key elements include:

- `id` is the deposition identifier, i.e. the bridge internal reference for depositions. It is used to access a specific deposition
- `status` is the deposition's status, as described above.
- `feeder_response` contains feedback from the processing of the deposition in feeder. The content of this field is also formatted in JSON. It is a black box from bridge's perspective. Upon deposition success, feeder will return a structure of the form:

```
{ "pids":
    [
      { "clientId":"c1", "pid":"CH-1234565-7:1"},
      { "clientId":"c2", "pid":"CH-1234565-7:2"},
         ...
    ],
  "feeder_version": "5.4.0"
}
```

It must be noted that:

- the `clientId` corresponds to the mandatory ids submitted by the client application for each object within the SIP (for example in the case of Docuteam DublinCore SIP it is located in dc.xml files and using the following syntax *<dc:identifier>clientId:d4FTw3v6T</dc:identifier>*).
- the `pid` is the persistent identifier allocated by the repository. It is of the form „{namespace}:{id}“, where „namespace“ is generally the institutional or ISIL code (for example: CH-1234565-7:2), and „id“ the unique number for that namespace in the repository.

Access API

This API is a proxy to docuteam rservices. rservices offers high-level access functionality to repository objects. For example, it is able to generate DIPs starting from any level of an archival package and assemble it recursively. Another notable feature is the on-the-fly generation of preview/thumbnail and, more generally, format migrations.

This access methods retrieve data from the repository, hence expects PIDs (and not bridge internal IDs, as it is the case for the depositions API).

In version 1.0, bridge is limited to synchronous requests, meaning that the required object is prepared and returned at once. Async/callback requests are not yet possible.

Routes

```
GET   /access/sync_metadata/:pid   sync_metadata#download
GET   /access/sync_dip/:pid        sync_dip#download
GET   /access/sync_original/:pid   sync_original#download
GET   /access/sync_preview/:pid    sync_preview#download
```

Metadata

Get the EAD metadata of a repository object.

Allowed calls

GET /access/sync_metadata/:pid

Requirements

Token with role `read`, `create`, `manage`, or `admin`.

Parameters

- `:pid` persistent identifier of a repository object
- `token`

Examples

```
curl
"https://bridge.docuteam.ch/access/sync_metadata/CH-123456-7:38?token=123456
789012345" --output ead.xml
```

Preview

Get a preview representation of a repository object.

Allowed calls

GET /access/sync_preview/:pid

Requirements

Token with role `read`, `create`, `manage`, or `admin`.

Parameters

- `:pid` persistent identifier of a repository object
- `token`

Examples

```
curl
"https://bridge.docuteam.ch/access/sync_preview/CH-123456-7:38?token=1234567
89012345" --output file.pdf
```

Original

Get the original format of a repository object.

Allowed calls

GET /access/sync_original/:pid

Requirements

Token with role `read`, `create`, `manage`, or `admin`.

Parameters

- `:pid` persistent identifier of a repository object
- `token`

Examples

```
curl
"https://bridge.docuteam.ch/access/sync_original/CH-123456-7:38?token=123456
789012345" --output file.pdf
```

DIP

Get the dissemination package of a repository object, structured according to the Matterhorn METS format. For nested objects, it is possible to receive a package with both binaries and (technical and descriptive) metadata recursively.

Allowed calls

GET /access/sync_dip/:pid

Requirements

Token with role `read`, `create`, `manage`, or `admin`.

Parameters

- `:pid` persistent identifier of a repository object
- `token`

Examples

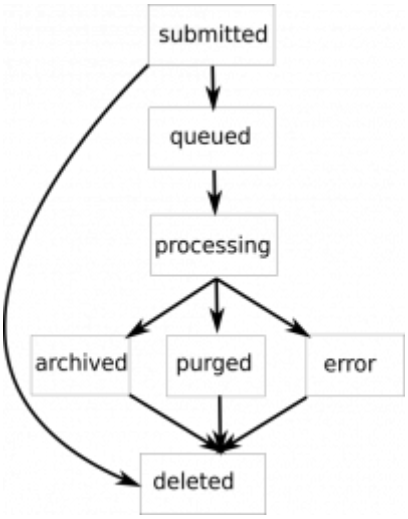
```
curl
"https://bridge.docuteam.ch/access/sync_dip/CH-123456-7:38?token=12345678901
2345" --output dip.zip
curl
"https://bridge.docuteam.ch/access/sync_dip/CH-123456-7:38?token=12345678901
2345&recursively=true  --output dip.zip
```

Changes API

Changes target specific objects in the repository, using their persistent identifier (PID), in order to replace or purge them. Similar to depositions, they need to be queued and processed by docuteam feeder to get their effect into the repository.

Status Model

- **submitted**: a new change corresponding to an update or purge was created in bridge
- **queued**: the change has been queued for processing in feeder
- **processing**: the change has been downloaded and is being processed by feeder
- **archived**: the change was successfully processed, i.e. the object in the repository was updated (depending on the repository setting this may create a new version of the object or replace it)
- **purged**: the change was successfully processed, i.e. the object was purged from the repository
- **error**: something went wrong, see the message in the „feeder_response“ field of the change
- **deleted**: the change was deleted from bridge



Routes

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

Create

Creates a new change.

Allowed calls

POST /changes

Requirements

Token with role `manage` or `admin`.

Parameters

- `pid` persistent identifier of a repository object
- `token`
- `task` ⇒ type of change, e.g. `data_update`, `metadata_update`, `object_update`, `data_delete`, or `object_delete`
- `package_format` ⇒ indicate the format of the submitted package, e.g. `MatterhornMets` or `DocuteamDublincore1.0` (optional, default: `MatterhornMets`)
- `[binary data]` ⇒ mandatory if task is „*_update“

Examples

```
curl -X POST -F "package=@sip.zip"
"https://bridge-stage.docuteam.ch/changes?token=123456789012345&pid=CH-654321-0:3&task=object_update&package_format=DocuteamDublincore1.0"
curl -X POST
"https://bridge-stage.docuteam.ch/changes?token=123456789012345&pid=CH-654321-0:3&task=object_delete"
```

Index

Lists the existing changes with details.

Allowed calls

GET /changes

Requirements

Token with role `manage` or `feeder`.

Parameters

- `token`
- `id` (optional)
- `status` (optional)
- `from` (optional, format:YYYY-MM-DD)
- `until` (optional, format:YYYY-MM-DD)
- `user` (optional)
- `organization` (optional)

Examples

```
curl "https://bridge.docuteam.ch/changes?token=123456789012345"
curl "https://bridge.docuteam.ch/changes?token=123456789012345&id=2"
curl
"https://bridge.docuteam.ch/changes?token=123456789012345&status=submitted"
curl
"https://bridge.docuteam.ch/changes?token=123456789012345&status=submitted&o
rganization=customerx"
curl
"https://bridge.docuteam.ch/changes?token=123456789012345&from=2018-11-01&un
til=2018-11-30"
curl
"https://bridge.docuteam.ch/changes?token=123456789012345&status=error&from=
2018-11-01"
```

Show

Retrieve the binary content of a change.

Allowed calls

GET /changes/:id

Requirements

Token with role `manage` or `admin`.

Parameters

- `:id` change ID
- `token`

Examples

```
curl "https://bridge.docuteam.ch/changes/1?token=123456789012345" --output
sip.zip
```

Update

Retrieve the binary content of a change.

Allowed calls

PUT /changes/:id

Requirements

Token with role `manage` or `admin`.

Parameters

- `:id` change ID
- `token`
- `status` ⇒ supported values are `data_update | metadata_update | object_update | data_delete | object_delete`
- `feeder_response` (optional, url encoded string)

Examples

```
curl -X PUT
"https://bridge.docuteam.ch/changes/12345?token=123456789012345&status=deleted"
curl -X PUT
"https://bridge.docuteam.ch/changes/23?token=12super34token56&status=archived&feeder_response=%7B%22pids%22%3A%5B%221%22%2C%222%22%5D%7D"
```

Responses

The responses on this API are similar to the depositions API responses, but have two additional fields:

- `pid` relates to the target repository object of the update.
- `task` describes the action to be performed (update, purge).

Practically, change responses look like this:

```
{ "api":
  { "name": "docuteam bridge",
    "version": "v1.0.0" },
"response":
  [
    { "id": 4321,
      "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": "myorganisation",
      "repository_key": "myrepository",
      "package_format" : "DocuteamDublinCore1.0",
      "package_attached" : true,
      "package_byte_size": 2716786,
      "task" : "node_update",
      "pid" : "CH-654321-0:87654"
     }
  ]
"request":
  { "organization": "myorganiztion",
    "role": "manage",
    "requested_at": "2018-11-03T11:13:39.278026Z"}
}
```