OTM Bridge API Specification

Notes/Assumptions

- An account will have a unique identifier or token within the Bridge that corresponds with a set of credentials that provide access to the Bridge API
- Each account will be created/initialized by the DDP operating the bridge and the credentials needed to access the account will be provided to the account owner
- There will be a one-to-one relationship between a Bridge account and a single repository (or other) system
- The repository (or other) system accessing the Bridge will have an internally consistent method for ensuring that File IDs provided to the bridge uniquely identify individual bitstreams presented to the bridge for storage

To Be Determined

- Method for authentication and authorization. Also need to determine if the same auth methods will be used by the Bridge and the Gateway

Bridge API

Bridge API - Endpoints for general use

Bridge Details

- Provides information about the bridge application. Can also be used to verify that the bridge application is available at the expected URL.
- Request: `GET /`
- Response Body: JSON

```
{
  "bridge-version" : "",
  "supported-checksum-types" : ""    # Possible values: MD5, SHA-256, SHA-512
}
```

- Response Code: 200 (on success)

Bridge API - Endpoints to be used by OTM Gateway

Register

- Purpose: Allows a repository with an OTM API to register itself with the Bridge. The information provided in this registration will allow the Bridge to make calls back to the OTM Gateway, such as to pull content listed in a deposit request. This call must be made prior to calls to deposit content. This call may also be made to update the Gateway information.

- Request: `POST /register`
- Request Body: JSON

```
{
  "gateway-url" : "",      # Endpoint URL where the Bridge can call back to this OTM API
  "gateway-username" : "", # Credentials to allow the Bridge to make calls back into the OTM API
  "gateway-password" : ""  # Credentials to allow the Bridge to make calls back into the OTM API
}
```

- Response Code: 200 (on success)
- Notes:
    - In the event that Gateway information is updated by calling this endpoint again, will those changes be applied to existing or in-process deposit requests?
    - In order to make this call, an account with associated credentials will already need to exist in the Bridge. This will have been created in advance by the DDP (see Add Account below).

Deposit Content

- Purpose: Allows the repository to send content to the Bridge for deposit into the DDP.
- Request: `POST /deposit ? {checksum-type}`
    - `checksum-type`: (Optional) Applies to all file checksums (can be one of: MD5, SHA-256, SHA-512). Default is MD5.
    - `deposit-format`: (Optional) Format of content being deposited
- Request Body: JSON

```
{
  "filegroup-1-id" : {                  # filegroup is a generic grouping of files that can be used to
capture structure such as in a digital object or work
    "version" : "",
    "files" : {
      "file-1-id" : "file-1-checksum",
      "file-2-id" : "file-2-checksum"
    }
  },
  "filegroup-2-id" : {
    "version" : "",
    "files" : {
      "file-3-id" : "file-3-checksum",
      "file-4-id" : "file-4-checksum"
    }
  }
}
```

- Response Code: 201 (on success)
- Notes:
    - Each filegroup in the request body is considered an independent deposit which can be referred to in subsequent requests by the filegroup identifier.
    - There is an explicit expectation that the content to be deposited will be available from the OTM Gateway at the path: "gateway-url" + / "filegroup-id" +/ + "file-id". How the Gateway resolves the file stream based on a request to this URL is irrelevant to the Bridge.
    - Provided Filegroup IDs and File IDs must be URL-safe to support the Bridge requesting those files from the OTM Gateway and the reverse in the restore context
    - There is no guarantee that all filegroups in a single deposit request will be deposited into the DDP at the same time. This allows the Bridge to manage transfers based on available resources (so as to not over-run local disk, for example).
    - The Bridge and DDP will consider the version value as opaque and not attempt to assign any meaning to the value.
        - Not including "version" in the JSON is acceptable and is functionally the same as "version" : "". This "empty" version is equivalent to any other version.
        - Attempting to deposit a filegroup with a filegroup ID and version that both match a previous deposit will be considered an error and rejected
        - When new filegroup versions are deposited, the Bridge may choose to retrieve and submit to the DDP only those files which have changed between versions (based on file ID and checksum).
    - The deposit-format parameter allows the depositor to specify the format of the content being deposited. The Bridge implementation may only support deposits of certain types.
        - Agreements between the depositor and the DDP should specify which types are to be used for deposit.

List Deposits

- Purpose: Retrieves a listing of all in-process deposits
- Request: `GET /deposit ? {status}`
    - `status:` (Optional) Limit list to deposits in a specific status
- Response Body: JSON

```
{
  "filegroup-1-id" : {
    "version : "",       # Version identifier of filegroup
    "files" : "",        # Number of files in deposit
    "status" : ""        # Current deposit status
  },                     # Additional filegroups listed here
}
```

- Response Code: 200 (on success)
- Notes:
  - List requests made by a depositor will return only those deposits which were initiated by that depositing entity
  - List requests made by the DDP (to determine if there are deposits which are ready for processing) will return results for all depositors
    - It may be useful to include a depositor query parameter to allow the DDP to limit the response based on depositor
  - This list will grow large if many deposits are initiated at once and may require paged results
  - This list will not include all historical deposits, but only those that are in the process of being deposited

Get Deposit Status

- Purpose: Allows the repository to ask for status of a given deposit
- Request: `GET /deposit/{filegroup-id}`
- Response Body: JSON

```
{
  "filegroup-id" : {
    "deposit-state" : "",
    "files" : {
      "file-1-id" : {
         "status",        # deposit status, one of: pending, retrieved, existing
         "size" : "",     # file size in bytes
         "checksum" : ""  # file checksum
       },
      "file-2-id" : {
         "status" : "",
         "size" : "",
         "checksum" : ""
       },
    }
  }
}
```

- Response Codes:
  - 200 (on success)
  - 404 (if the provided Deposit ID does not exist in the system)
- Notes:
  - It would be possible to retain enough information to be able to provide a tombstone-type response for filegroups which have completed deposit.
    - Alternatively, requests for filegroups that have completed deposit could be forwarded to the Get Deposited endpoint.
  - Deposit status
    - Pending - File is waiting to be processed (it has not been retrieved)
    - Retrieved - File has been successfully transferred to storage manage by the bridge
    - Existing - File with a matching ID and checksum was part of a previous version that is already stored; this file will not be retrieved

Delete Content

- Purpose: Removes one or more files from DDP storage
- Request: `POST /delete ? {checksum-type}` # Using POST rather than DELETE, allows for deleting multiple files (and is consistent with Restore action)
  - `checksum-type`: (Optional) if provided, applies to all file checksums (can be one of: MD5, SHA-256, SHA-512). Default is MD5.
- Request Body: JSON

```
{
  "filegroup-1-id" : {
    "version" : "",
    "files" : {
      "file-1-id" : "file-1-checksum",     # Checksum is optional, can be included to verify correct
file is being deleted
      "file-2-id" : "file-2-checksum"
    }
  },                                    # Additional filegroups listed here
}
```

- Response Code: 202 (on success)
- Response Body: JSON

```
{
  "delete-id" : ""
}
```

- Notes:
  - It would be possible to retain enough information to be able to provide a tombstone-type response for File IDs which were at some point in the system but have been removed.
  - It would be possible to allow the "files" section to be omitted if all files in a given filegroup version are to be deleted.
  - It would be possible to allow the value associated with a filegroup ID to be omitted if all files in all versions of a filegroup are to be restored.

List Deletes

- Purpose: Retrieves a listing of all in-process deletes
- Request: `GET /delete ? {status}`
  - `status:` (Optional) Limit list to deletes actions with a specific status
- Response Body: JSON

```
{
  "delete-id-1" : {
    "files" : "",     # Number of files in delete action
    "status" : ""     # Current delete status
  },                  # Additional delete actions listed here
}
```

- Response Code: 200 (on success)
- Notes:
  - Used by both the OTM Gateway (to check on delete requests) and the DDP (to determine if there are delete requests which are ready for processing)
  - Delete IDs returned by this method are only those that are in-process.

Get Delete Status

- Purpose: Allows the repository to ask for status of a content delete action
- Request: `GET /delete/{delete-id}`
- Response Body: JSON

```
{ "status" : "" }   # This could include a top level status or a per-file status (or both)
```

- Response Code: 200 (on success)
- Notes:
  - Need to better define the information that would be provided in the status response

List Deposited

- Purpose: Retrieves a list all deposited filegroup IDs
- Request: `GET /list`
- Response Body: JSON

```
{ "filegroup-1-id", "filegroup-2-id", ... }
```

- Notes:
    - This list would not include deleted filegroups

Get Deposited Details

- Purpose: Retrieves details about a deposited filesgroup, including versions and associated files
- Request: `GET /list/{filegroup-id}/{file-id}`
    - `filegroup-id` : The identifier of the filegroup for which information is requested
    - `file-id` : (Optional) The identifier of the file for which information is requested
- Response Body: JSON

```
{
  "checksum-type" : ""      # Indicates type of checksums listed below, can be one of: MD5, SHA-256, SHA-
512
  "filegroup-1-id" : [      # filegroup is a generic grouping of files that can be used to capture
structure such as in a digital object or work
    {
      "version" : "",       # filegroup version
      "files" : {
        "file-1-id" : {
           "size" : "",     # file size in bytes
           "checksum" : ""  # file checksum
         },
        "file-2-id" : {
           "size" : "",     # file size in bytes
           "checksum" : ""  # file checksum
         },                 # Additional files listed here
      }
    },                      # Additional filegroup versions listed here
  ]
}
```

- Notes:
    - When a file-id is included, the file matching that ID is the only one returned in the response
    - This list would not include deleted versions or files. A tombstone capability is possible, but not currently part of this specification.

Restore Content

- Purpose: Requests that content be made available for restore
- Request: `POST /restore ? {checksum-type}`
    - `checksum-type`: (Optional) if provided, applies to all file checksums (can be one of: MD5, SHA-256, SHA-512). Default is MD5.
- Request Body: JSON

```
{
  "filegroup-1-id" : {
    "version" : "",
    "files: {
      "file-1-id" : "file-1-checksum",     # Checksum is optional, can be included to verify correct
file is being restored
      "file-2-id" : "file-2-checksum"
    }
  },                                    # Additional filegroups listed here
}
```

- Response Code: 202 (on success)
- Response Body: JSON

```
{
  "restore-id" : ""
}
```

- Notes:
    - In the versioning scenario, checksum may be used to select which version to restore. If no checksum is provided most recent version is assumed.
    - It would be possible to allow the "files" section to be omitted if all files in a given filegroup version are to be restored.

- It would be possible to allow the value associated with a filegroup ID to be omitted if all files in all versions of a filegroup are to be restored.

List Restores

- Purpose: Retrieves a listing of all in-process restores
- Request: `GET /restore ? {status}`
  - `status:` (Optional) Limit list to restores actions with a specific status
- Response Body: JSON

```
{
  "restore-id-1" : {
    "files" : "",      # List of files in restore action
    "status" : "",     # Current restore status
    "expiration" : ""  # Date on which restored content will no longer be available
  },                   # Additional restore actions listed here
}
```

- Response Code: 200 (on success)
- Notes:
  - Used by both the OTM Gateway (to check on restore requests) and the DDP (to determine if there are restore requests which are ready for processing)
  - Restore IDs returned by this method are only those that are in-process.

Restore Status

- Purpose: Allows the repository to ask for status of a specific content restore action
- Request: `GET /restore/{restore-id}`
- Response Body: JSON

```
{
  "files" : "",      # List of files in restore action
  "status" : ""      # This could include a top level status or a per-file status (or both)
  "expiration" : ""  # Date on which restored content will no longer be available
}
```

- Response Code: 200 (on success)
- Notes:
  - Restored content will only be made available on the Bridge for a limited time. After this time, content will be removed and the status of the restore will be changed to expired.

Get Restored Content

- Purpose: Allows the repository to retrieve restored content from the Bridge
- Request: `GET /restore/{restore-id}/{filegroup-id}/{file-id} ? {checksum-type}`
  - `filegroup-id:` The identifier of the filegroup for the restored file
  - `file-id:` The identifier of the restored file to retrieve
  - `checksum-type`: (Optional) Defines the type of checksum to be included in the response ETag header. Can be one of: MD5, SHA-256, SHA-512. Default is MD5.
- Response: Restored file content stream
- Response Codes:
  - 200 (on success)
  - 404 (if the file is not part of the restore or the restored content has expired)
- Response Headers: ETag

Get Audit Log

- Purpose: Retrieves an audit history report. The report will list audit events associated with a single file or set of files in a filegroup.
- Request: `GET /audit/{filegroup-id}/{file-id}`
  - `filegroup-id:` The identifier of the filegroup for which an audit report is requested
  - `file-id:` (Optional) The identifier of the file for which an audit report is requested. If this is not provided the audit report is requested based on the filegroup-id.
- Response Codes:
  - 200 (on success)
  - 404 (if there is no audit information for the provided ID)
- Response Body: JSON

```
{
  "filegroup-id" : [
    {
```

```
      "file-id" : [
        {
          "date" : "",
          "type" : "",
          "event" : ""
        },                # Additional events for this file listed here
      ],                  # Additional files listed here
    }
  ]
}
```

- Notes:
  - It may be desirable to provide paged results for large result sets
  - Types and fields of expected history events are to be listed in an appendix

Bridge API - Endpoints to be used by DDP

Add Account

- Purpose: Allows the DDP to create a new depositing user account in the Bridge. After this call it will be possible for the OTM Gateway to register with the Bridge. This call can also be used to reset the access credentials for an account.
- Request: `POST /account/{account-name}`
  - `account-name:` The name associated with an account
- Response Code: 201 (on success)
- Response Body: JSON

```
{
  "account-name" : "",      # Name of the account
  "account-username" : "",  # Credentials to allow calls to the Bridge for this account
  "account-password" : ""   # Credentials to allow calls to the Bridge for this account
}
```

List Accounts

- Purpose: Allows the DDP to list all accounts known by the Bridge
- Request: `GET /account`
- Response Code: 200 (on success)
- Response Body: JSON

```
{ "account-1-name", "account-2-name" }
```

Complete Deposit

- Purpose: Allows the DDP to inform the Bridge that a deposit has completed.
- Request: `POST /deposit/{filegroup-id}`
- Response Code: 200 (on success)

Complete Delete

- Purpose: Allows the DDP to inform the Bridge that a delete has completed.
- Request: `POST /delete/{delete-id}`
- Response Code: 200 (on success)

Complete Restore

- Purpose: Allows the DDP to inform the Bridge that a restore has completed.
- Request: `POST /restore/{restore-id}`
- Response Code: 200 (on success)

Add Audit Event

- Purpose: Update audit index by providing additional audit details
- Request: `POST /audit/{id}`
  - `id:` The identifier for the file or filegroup to which this audit event is to be applied
- Request Body: JSON

```
{ "audit-event" }
```

- Response Code: 200 (on success)
- Notes:
    - Rather than a direct call from the DDP to the Bridge, this could be implemented as a notification stream to which the Bridge is listening.