Minutes of the HTTP REST TAPE API meeting (16th of April 2021)

Goals we wanted to achieve for this meeting

- Gather dCache, StoRM, EOSCTA use cases for the **version 1** of this tape REST API
- Decide on a way to communicate among ourselves

Document we used to start our discussions

https://docs.google.com/document/d/1xioJmM1cr9iWaTd-8cpM7f6h3wP4qNfvAGxcHWSvOQs/edit#heading=h.uo4rpbytmgqm

Minimum functionalities we would like to have in common

Here are the minimum functionalities we would like to have in common for this first version of the API:

- Submit bulk “stage” requests: give a list of files to the API to trigger their retrieval from tape
- Cancel “stage” requests: a user can cancel the each retrieve request in the bulk request
- Evict disk copy from the cache.
- Get informations about files known by the system (file archival use case)
- Delete bulk requests (cancel + clear)
- Define the lifetime of a bulk request
- Limit the number of bulk-requests submitted by users
- Limit the number of files contained in a bulk-request

REST API implementation to provide these functionalities

Submit bulk “stage” requests: give a list of files to the API to trigger their retrieval from tape

This will be done by issuing a `STAGE` activity on the resource `api/v1/bulk-requests` (POST).

The `STAGE` activity allows to stage all the files located in the bulk request.

This activity was “invented” during this meeting. Indeed, even if the `PIN (lifetime=0)` would

do the job, the main goal of the `PIN` activity is to be able to give the user the possibility to let a file on disk for a certain amount of time (retention policy).

To be decided in the next meeting: what are the messages (json) that will be exchanged between the client and the server ?

Cancel “stage” requests: a user can cancel the each retrieve request in the bulk request

This will be done by issueing an HTTP PATCH command on the resource `api/v1/bulk-requests/id`.

To be decided in the next meeting: what are the messages (json) that will be exchanged between the client and the server ?

Evict disk copy from the cache

This will be done by issueing an `UNPIN` activity on the resource `api/v1/bulk-requests` (POST).

The `UNPIN` activity will allow to evict the disk copies of all files in the bulk request.

To be decided in the next meeting: what are the messages (json) that will be exchanged between the client and the server ?

- List of files from the client

Get informations about files known by the system

One use-case for this functionality is to allow to track the progress of files archival to tape.

This will be achieved by issueing a `FILEINFO` activity on the resource `api/v1/bulk-requests` (POST).

Here is the list of information we would like to have per file in the bulk request. The list will be extended during the next meeting:

- Disk residency
- Tape residency
- File existence in the namespace
- errors
- checksum

For simplicity reasons, we would like this `FILEINFO` activity to be asynchronous like the other ones. Also, on dCache, getting these file informations might take some time.

To be decided in the next meeting: what are the messages (json) that will be exchanged between the client and the server ?

Delete a bulk request

This will be done by issuing an HTTP `DELETE` command on the resource `api/v1/bulk-requests/id`. This will have the effect of `CANCEL` + `CLEAR` the bulk request.

To be decided in the next meeting: what are the messages (json) that will be exchanged between the client and the server ?

Define the lifetime of a bulk-request

To be decided in the next meeting: how do we define the lifetime of a `bulk-request` ?

Is it a constraint we want do expose ? If yes how do we show this information to the user ?

Limit the number of bulk-requests submitted by users

To be decided in the next meeting: how do we limit the number of bulk-requests submitted by the users ?

Is it a constraint we want do expose ? If yes how do we show this information to the user ?

Limit the number of files contained in a bulk-requests

To be decided in the next meeting: how do we limit the number of files a user can submit per-bulk request.

Is it a constraint we want do expose ? If yes how do we show this information to the user ?

Summary

Functionality	Implementation
Submit bulk “stage” requests	POST `api/v1/bulk-requests`. Activity=`STAGE`
Cancel “stage” requests	PATCH `api/v1/bulk-requests/id`
Evict disk copy from the cache	POST `api/v1/bulk-requests`. Activity=`UNPIN`
Get informations about files known by the system (file archival tracking use case)	POST `api/v1/bulk-requests`. Activity=`FILEINFO`
Delete bulk requests (cancel + clear)	DELETE `api/v1/bulk-requests/id`
Define the lifetime of a bulk request	To be decided
Limit the number of bulk-requests submitted by users	To be decided
Limit the number of files contained in a bulk-request	To be decided

What needs to be discussed in a future meeting

- Message exchanges between the client and the server for each functionality. Define the JSON structure of the request/response, ERROR codes when the operation is not supported, etc…
- What is going to be returned, per file, by the `FILEINFO` activity
- API Discovery mechanism
- Per-VO limitations ?
- Bulk-requests limitations
- Authentication users and server can interract with our API, how do we decide to authenticate ?
- Exposure of the API limitations/constraints to the user ?

Ways of communication

- Cedric will create an e-group where anyone who is interested can subscribe to. egroup name: `wlcg-tape-rest-api-discussions`
- We will report at each DOMA-TPC meeting our progress on this API.

We will organize a future meeting soon to continue the discussion.

Food for thought for the version 2 of the API

We have discussed ideas for the version 2, I just list them so that we do not forget about them.

- Support for synchronous operations
- PAUSE, UNPAUSE bulk-requests