dCache.org

dCache bulk requests

Paul Millar
on behalf of the dCache team

DOMA-TPC
2021-02-10
https://indico.cern.ch/event/1006673/

Nordic e-Infrastructure Collaboration

Fermilab

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

DESY.

dCache REST API: why and what

- Avoid non-standard protocol extensions:
  - Avoid shoe-horning dCache concepts into existing protocols
  - "dot commands" in NFS, POST requests in WebDAV, the -X "magic" in dcap commands, SITE commands in FTP, …
- Follow **best practice** for REST APIs.
- No direct data access; use WebDAV (…) instead.
- **Well documented**: both machine and human readable
- Seems to be **well-received**:

  JavaScript client. Sites started integrating it into their tools.

Limitation with a REST API

- Best practice: **URL** defines the resource on which to operate.
- HTTP requests can have **only one** URL.
- Single file requests are a **major bottleneck**:
  - SRM supports bulk operations, for very good reasons.
  - For example, SRM is orders-of-magnitude faster at deleting files than any single-file protocol alternative.
  - CTA came up with an (undocumented) extension that adds bulk requests to the xroot protocol.
- So, we need to **“fix” this** for REST API.

REST Bulk request API

- **Extension** to existing REST API
  - Do not break existing clients.
- Protocol **documented** in a Google docs.
  - This describes client interactions: what the client can expect: user interactions and a processing model.
  - It does NOT describe how this is implemented.
  - https://docs.google.com/document/d/14sdrRmJts5JYBFKSvedKCxT1tcrWtWchR-PJhxdunT8/edit?usp=sharing
- (Again) Try to follow **best practice**.
- Try to fold in our **experience** from SRM.

REST Bulk: core concepts

- **Bulk operation**:

  the general idea of asking that something happens to multiple files stored in dCache.

- **Activity**:

  a simple, well-defined operation that affects files individually (e.g., stage, pin, delete) or a set of files collectively (e.g., create concatenated file, create an archive).

- **Bulk request**:

  A specific request to dCache make by a specific user that targets a specific set of files, with a specific activity.

REST Bulk: goals

It should be possible to …

- target an **arbitrary list of files**.
- target a directory and everything in it (direct children or recursively).
- **extend** the list of supported activities without breaking existing clients.
- make **other requests** while a bulk request is processed.
- **check the progress** of a bulk request.
- **cancel** a bulk request.
- learn the **final result** of a request, even when cancelled.
- **discover** all on-going and completed bulk requests.

REST Bulk: non-goals

- The API does not allow you to …
  - make a bulk request with **multiple activities**.
  - cancel only **part** of a bulk request.
  - **Revert/undo** the activity when cancelling a bulk request.
  - receive **asynchronous notification** … (at least, not yet 😄)

REST Bulk: core features

- Flexible **target** selection:
  - List of targets, directories with children, fully recursive
- **Separation** between request cancellation and request clearing.
- Flexible **auto-clearing** of requests
- Request **introspection**: target selection, progress, errors so far, original request
- Request **discovery**:
  - see all uncleared bulk requests,
  - Filter to show only certain requests (e.g., incomplete).

REST Bulk: whistle-stop tour

Create a request:

POST request (with JSON object) targeting `/api/v1/bulk-requests`

If successful, response contains Location header, with ID for that request (a URL)

Current status:

- `GET` request targeting <ID>
- Response is JSON-Object

Cancel a request:

- `PATCH` request (w/ JSON object) targeting <ID>

Clear a request:

- `DELETE` request targeting <ID>
- Clearing an on-going request also cancels it.

REST Bulk: current status

- Initial version available with (not yet released) dCache v7.0
- Supports placing limitations on number of bulk requests:
  - May define limits per-user and total.
- Supported activities:

  `PIN`, `UNPIN`, `UPDATE-QOS` and `DELETE`
- Future work: stress testing and feedback from production use.

Connection with DOMA-TPC

- Start of a **standardisation effort**:
  - Bulk API considered a starting point for a storage agnostic tape interface.
  - So far, all interested parties seem somewhat agreeable.
  - Very early days .. need feedback from other developers.
- **Clients** would also need to be written
  - Perhaps similar to how SRM+GridFTP and SRM+HTTP-TPC work currently.
- The API is **decoupled** from the actual data operation:
  - FTS could do Bulk-API+GridFTP or Bulk-API+HTTP-TPC or Bulk-API+xroot-TPC.

Open issue #1: endpoint discovery

- In dCache, REST API (and so, Bulk API) is a separate endpoint:

  In particular, REST API requests don't work against the WebDAV endpoint.
- Should CRIC record two endpoints: REST API + data-transfer endpoint (e.g., WebDAV)?
- If CRIC store just one endpoint, how is the other endpoint discovered?
  - Both endpoints are the same (break clean separation)
  - Add discovery mechanism (WebDAV → REST or vice versa)
  - dCache REST already supports WebDAV discovery.

Open issue #2: writing to tape.

- So far, talking about staging data back from tape.
- What about uploading data so it's written to tape?
- Several options, including …
  - Use Bulk API's **QoS transition**, after upload completes.
  - Something in the data-transport protocol; e.g., an HTTP header in PUT request (or something in the URL).

    (Would already work with HTTP-TPC.)
  - Use path to identity target media.

Thanks for listening