Direct DOI Data Access (D3A)

Rolf Krahl

ICAT F2F Meeting, 10 February 2026, Abingdon, UK

Problem Statement

- DOIs (and PIDs in general) for data usually resolve to a landing page, containing informations on how to download the data.
- This works fairly well for humans, but not at all for machines.
- The LEAPS WG3 Special Interest Group "D3A" is working on a common approach to make the access to the data machine actionable.
- That means: a computer program, taking a PID as input, should be able to deliver the downloaded data as output.
- This problem can be split into two sub problems:

  where from the PID, get the location, e.g. a download URL for the data,

  how given the URL, do the download (also if the data is on tape).

Proposed Solution for "where": Metalink

- Metalink is a XML document format that describes a file or list of files to be downloaded from a server.
- It contains metadata about the files (description, hashes) and a list of URLs to download them.
- Metalink is an IETF standard: RFC 5854
- The landing page for a dataset could offer a Metalink document via content negotiation (`application/metalink4+xml`).

Implementation Notes for the Metalink Solution

- I intend to implement a tiny "permalink" web service script as a separate component for ICAT.
- It should implement "HTTP GET" for Investigations and Datasets in ICAT. It should take appropriate attributes of the objects as path parameter: (`investigation.name`, `investigation.doi`, `dataset.name`, ...: tbd)
- It should implement content negotiation for (at least) the following media types:

  `text/html` redirect to a human readable landing page or a deep link into the ICAT web user interface,

  `application/vnd.datacite.datacite+xml` send DataCite bibliographic metadata describing the data object,

  `application/vnd.schemaorg.ld+json` send schema.org bibliographic metadata describing the data object,

  `application/metalink4+xml` send a Metalink document.

Implementation Notes for the Metalink Solution (cont.)

- The "permalink" web service script would also implement download links for the files.
- These download links would make a `prepareData()` call to `ids.server` and redirect to the corresponding download link based on the `preparedId`.
- The PIDs for the raw data from HZB will then resolve to this "permalink" web service script.
- The data publication landing pages will have the same functionality built in.

Discussion for "how"

- If the data is on disk, the "how" is trivial: just download it.
- If the data is on tape, any HTTP download request will time out before the download could possibly start.
- Automated download of data on tape could work as follows:
  1. The client requests the download from the given URL,
  2. The server replies: "Sorry, that data is currently not available, please come back later,"
  3. At the same time, the servers triggers the staging of the data from tape to disk,
  4. The client retries, after an appropriate delay, using the same URL,
  5. Now the data is on disk, download succeeds.
- Incidentally, this is exactly what `ids.server` already implements.

Discussion for "how" (cont.)

- Only problem here: how to signal that "Sorry, please come back later" in a way that the client understands?
- We would need an appropriate HTTP Status Code for that.
- In the absence of anything better, `ids.server` uses 503 "Service unavailable". That is the closest pick from all existing status codes, but maybe still not universally understood in the way we intend.
- We would need something like 5XX "Requested resource currently not available" with the defined semantic that the client should retry the same request again later.
- Maybe establish a custom 5XX status code as a community standard?
- Maybe even think big: propose a new 5XX status code as a universal standard?