GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Persistent TAP Uploads in DaCHS 2.11

Markus Demleitner

IVOA Southern Spring Interop 2024, Malta Nov 14-17
DAL WG

Outline

- Persistent uploads?
- Table PUT, GET, and DELETE
- Metadata retrival
- CREATE TABLE
- Discovering persistent upload facilities
- Open questions

Persistent Uploads

TAP has always supported table uploads.

But tables are gone after a request, requiring re-uploads, making reuse difficult and inefficient.

So, let's give TAP users a facility to put tables into the server-side database across requests.

Prior Art

- CADC has youcat; see Pat's 2018 Interop talk: PUT-s on the VOSI tables endpoint.
- ESAC has VOSpace-based table uploads, coming with lots of Authz.

The present effort follows youcat somewhat, but does not overload VOSI tables.

PUT, GET, DELETE

- To upload a table, do an HTTP PUT with a VOTable payload to `user_tables/<table-name>`
- To retrieve the server-side metadata, do an HTTP GET on `user_tables/<table-name>`
- To delete a persistent table, do an HTTP DELETE to `user_tables/<table-name>`

Slight Trouble: What to Return?

This is admittedly somewhat flamboyant right now.

- PUT returns an informative text/plain string. Redirect to the table metadata instead?
- GET returns a VOSI table. That's sane.
- DELETE returns an informative text/plain string.

Errors are communicated as DALI-compliant VOTables.

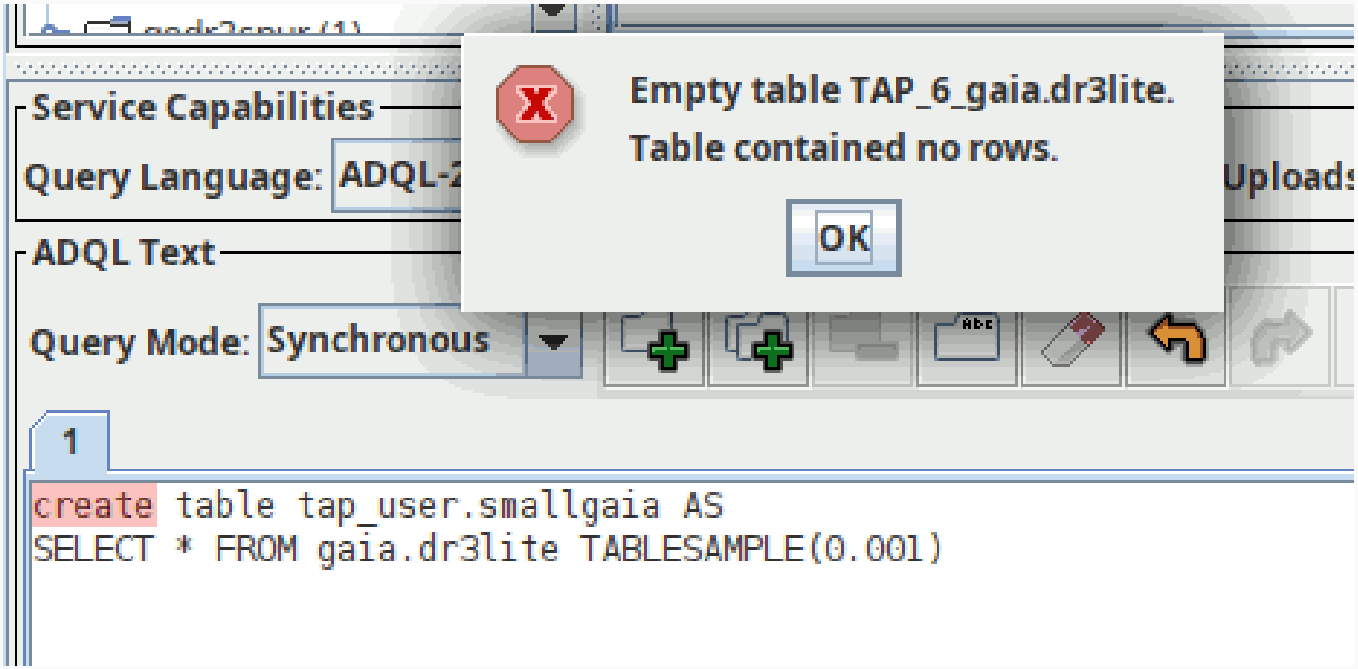
Metadata Retrieval

A GET on `user_tables` returns a VOSI tableset for all uploaded tables. No support for DETAIL=min in DaCHS so far, though.

Except if you support anonymous persistent uploads (DaCHS does): there is no way to find these once you have forgotten your table name.

CREATE TABLE

I have also made an ADQL extension:



These queries return the result of the query without rows and create a new persistent table with the rows.

Discovering Persistent Table Facilities

How do you find out whether a given TAP service supports persistent uploads?

Mark Taylor on the DAL list suggested to do a GET against `user_tables`. Then:

- 404: unsupported
- 403: supported (but you're not authenticated)
- 200: supported (and you are authenticated)

This does not help to figure out whether you can upload without auth. TAPRegExt? We could easily define a feature for that.

Open Questions

- How to prolong the life time of an uploaded table (current default: one week)?
- How to create indexes on them (in particular spatial ones)?
- Authz: Do we want to make these tables shareable between different users? [full disclosure: I don't]
- Scaling: DaCHS has no quota on these yet, but in production we would probably need some. How do people discover those?

Try it!

There's curl calls and a jupyter notebook in our blog post on this:
https://blog.g-vo.org/a-proposal-for-persistent-tap-uploads.html

If you are running DaCHS, upgrading to 2.10.2 will let you play with this on your own server.