GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Finally: Towards TAPRegExt 1.1

Markus Demleitner

Astro-CC Technology Forum 1, 2025-11-08

TAPRegExt 1.1

- What's TAPRegExt anyway?
- Per-mode limits: limits groups. . .
- . . . or forMode attributes?
- Is this where DALIInterface belongs?
- Deprecate dataModel?

What is TAPRegExt?

The TAP specifiction has many should-s and may-s, e.g.,

- may support different languages
- languages have optional features
- should support uploads (with various methods)
- may kill queries after a while
- may truncate results

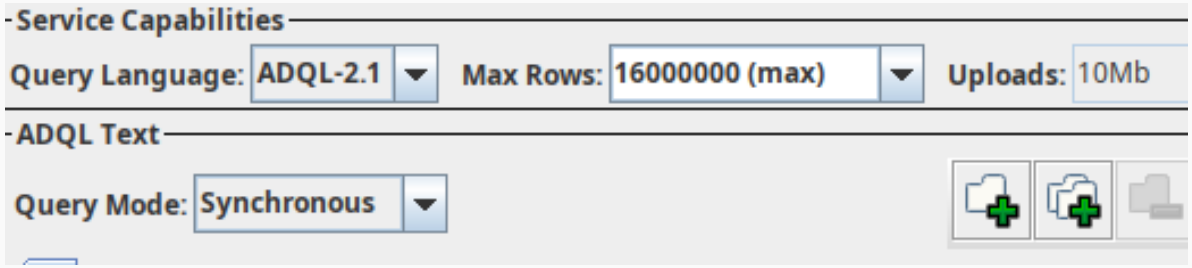
TAPRegExt is an XML schema that lets TAP services communicate their choices to clients machine-readably.

Per-Mode Declarations

TAP lets you do sync ("do it now") or async ("feel free to queue") queries.

Some services enforce different limits depending on this "mode".

E.g., Heidelberg's DaCHS: 10 seconds of runtime on sync, default 7200 seconds and unlimited on async.



Per-Mode-Limits: First Attempt

This started in pyVO bug #685, where Stelios wanted to say "use async on my service".

First design inspired by that: limit groups depending on "mode", which includes things like anon-sync and auth-async:

```
<outputLimit>
  <default unit="row">20000</default>
  <hard unit="row">20000000</hard>
</outputLimit>
<uploadLimit>
  <hard unit="byte">10000000</hard>
</uploadLimit>
```

```
<limits mode="anonymous-async">
  <executionDuration>
    <default>7200</default>
  </executionDuration>
  <uploadLimit>
    <hard unit="byte">80000000</hard>
  </uploadLimit>
</limits>
```

This is still in TAPRegExt PR #8.

Per-Mode Features: Second Attempt

In the discussion on PR #8, a (probably better) alternative materialised: a forMode attribute on outputFormat, uploadMethod, retentionPeriod, executionDuration, outputLimit, and uploadLimit.

forMode can be sync and async; auth is orthogonal: serve capabilities for the user you see.

```
<executionDuration>
  <default>50</default>
</executionDuration>
<executionDuration forMode="async">
  <default>7200</default>
</executionDuration>
```

```
<outputFormat forMode="async">
  <mime>application/parquet</mime>
</outputFormat>
```

Per-Mode Features

Right now, nothing in the vicinity of language (including UDFs and all) has a mode attribute.

I think I wouldn't even want to enable that.

And auth is orthogonal to forMode anyway (in case you think about only allowing certain languages to authenticated users).

Why DALIInterface?

TAP services are still registered with their base URL in a vs:ParamHTTP interface. Everything about that as wrong.

This results in serious pain. See http://ivoa.net/documents/caproles/.

To fix this, we need a suitable interface type. In 2019, I proposed DALIInterface for that.

I'd still like to have that.

DALIInterface for TAP

```
<capability standardID="ivo://ivoa.net/TAP">
  <interface xsi:type="vs:DALIInterface">
    <accessURL>http://example.org/tap</accessURL>
    <mirrorURL>http://example.com/tap</mirrorURL>
    <endpoint name="async"/>
    <endpoint name="sync"/>
    <endpoint name="tables"/>
    <endpoint name="capabilities"/>
    <endpoint name="examples"/>
  </interface>
</capability>
```

(we'd still have a vs:ParamHTTP in TAPRegExt 1.1 for backwards compatibility).

DALIInterface Extensibility

Perhaps the most controversial idea with DALIInterface: I'd like to let people attach RDF triples to endpoints using RDFa attributes:

```
<endpoint name="async">
  <meta property="http://ivoa.net/rdf/tap#hasArchived" resource="1"/>
  <meta property="http://example.edu/bibvoc#accesses"
    resource="https://astro-cloud.edu/vospace"/>
</endpoint>
```

(you could have `about`, too, in case you want a subject different from the endpoint).

Deprecate dataModel

dataModel is another thing we've been doing wrong: adherence to a data model is a property of tables or schemas, not of TAP services.

This is fixed in EPN-TAP and ObslocTAP, RegTAP is moving, Obscore needs to move.

See http://ivoa.net/documents/Notes/TableReg.

I'd like to go ahead and deprecate dataModel so people don't fall into that trap again.

Implementation Status

- Per-mode limits: DaCHS can produce these, but dc.g-vo.org doesn't to avoid ugly pyVO warnings on registry queries. Talk to me if you need a testing system.
- DALIInterface: DaCHS can produce it but doesn't anywhere in production
- Client side: Nothing yet.

Opinions?

- Do you find merit in the limits element?
- Do you find fault with forMode?
- Are there modes other than sync and async?
- How scared are you of DALIInterface?
- Should it go to DALI in the first place?
- How scary is RDFa in VOResource to you?
- Would you like to wait with deprecating dataModel?