Fig. 1

Fig. 2

Fig. 3

1. Plans for VODataService 1.2

(cf. Fig. 1)

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(cf. Fig. 2)

- VODataService 1.1 is older than TAP 1.0
- VOSI, Discovering Data Collections want changes for consistency
- STC-X for coverage didn't work.
- Misc requirements from VODataServiceNext on twiki

So – what's on the table already?

(cf. Fig. 3)

2. Improving coverage

See separate talk

3. DALI-typed interfaces

See separate talk

4. Capabilities on vs:DataCollection

vs:DataCollection's content model currently is:

. . . facility*, instrument*, rights*, format*, coverage?, tableset?, accessURL?

accessURL would be a link to a web page.

Better model: have capabilities.

- aux capabilities for discovering data collections
- generic capabilities with WebBrowser interfaces for what accessURL was for (permits mirrors, alternative access protocols. . .)

The downside is that the difference between DataCollection and CatalogService becomes tenous indeed. But that's not much of problem.

5. Schema-less Tables

There are many TAP services publishing tables to be used without schema qualification.

Right now, people hack around it using "public" in the tableset:

```
<vosi:tableset>
  <schema>
    <name>public</name>
    <description>Default schema</description>
```

Null information like this is a danger: Can we get rid of it?

- Tables as direct children of tableset? We already had that in VODataService 1.0; it will cost us the foreign key from tables to schemas in RegTAP, though.
- Allow empty name and description? That's still fairly ugly in my book but is less intrusive.

Note that the original idea – that people build names using the schema name – doesn't work in practice for several reasons. Schema metadata in the tableset thus isn't used much (it is in the GAVO DC, though).

6. Table Names

In VOSI and TAP_SCHEMA, table/name has to be given "ADQL-ready", i.e., with schema and any delimiters required (typically, double quotes).

The double quotes would look a bit odd in generic tablesets.

- Insist on consistency between VOSI and TAP_SCHEMA?
- Think of something else? Frankly, the thought of what horrors might lay that way makes me shudder.

7. Richer BaseParam Metadata

vs:TableParam content model:

name?, description?, unit?, ucd?, utype?, dataType?, flag*

Versus VOTable, that's missing VALUES (i.e., enumeration of legal values, ranges) – which is instrumental when building UIs, e.g., from VOSI capabilities.

There's discovery cases, too: "Services publishing data where R-band photometry goes below 20 mag", "Services accepting ABSOLUTE for CALIBRATION". But of course it would take fairly high takeup of domain matadata declaration before this would really be a good idea.

- Guidance : VOTable PARAM content model
- Histograms? Null ratios? For instance, http://gaia.ari.uni-heidelberg.de/tap/tables has Mean, Stddev, Median, Quartiles, histograms for Gaia tables. Such metadata would be necessary for cross-service query planning.

8. Richer Table Metadata

VOSI tables providers have asked for

- number of rows
- coverage (but: resource metadata?)
- DOI (but: resource metadata?)

To me, size is reasonable; beyond that: We shouldn't step into a resource-within-resource trap.

9. vs:ParamHTTP return type

ParamHTTP (and perhaps others) resultType is 0...1.

With DALI RESPONSEFORMAT, that would need to be 1..n.

Alternatives: Deprecate resultType and...

- ... adopt TAPRegExt's outputFormat?
- ... rely on rich metadata on a RESPONSEFORMAT declaration?
- ... forget about it? This last option seems attractive because at this point I don't know any use case for returnType.

10. dataproduct_types

As more and more ObsTAP services come online, it'd be great if users looking for, say, time series could pre-select only those services that actually have time series.

Have this (or any other obscore concept) in the DataService metadata?

Of course, VALUES in tableset columns would work as well.

11. Just One Type System

VODataService 1.1 has

- vs:SimpleDataType (integer, real, complex, boolean, char, string)
- vs:TAPType (BOOLEAN, SMALLINT, INTEGER, BIGINT, REAL, DOUBLE, ...)
- vs:VOTableType

This just makes things complicated for everyone. Let's do like TAP 1.1 has: Just make everything vs:VOTableType

12. Work Plan

- Early July 2018: Translation of HTML VODataService 1.1 source to RegTAP (me)
- August 2018: Introducing agreed-upon features (Volunteers?)
- September 2018: First internal working draft (Volunteer?)
- November 2018: Next iteration at Interop