1. DataCollections in Use

Markus Demleitner, gavo@ari.uni-heidelberg.de

We can register data collections in the VO. With TAP, ObsCore, and to keep the number of typed services reasonable, we should do so.

- Scenarios
- In VOResource
- The Server Side
- Thoughts on a User Interface

2. Scenario for DataCollections I

A SIAP service providing images of lensed quasars from multiple observatories.

I want

- one SIAP service to
  - keep all-VO searches feasible and to
  - provide one-stop shopping for the lensing community.
- *and* the metadata for each data collection (observatory, instrument, coverage, creator, etc) for
  - discoverability
  - provenance
  - satisfying the data provider's vanity

3. Scenario for DataCollections II

A TAP service giving access to many tables.

I want

- a single TAP service giving access to all data to
  - allow joins without large overhead
  - keep all-VO TAP searches (e.g., ObsTAP) feasible
- *and* records for all the individual tables in the registry (reasons see above).

4. ...in VOResource, data

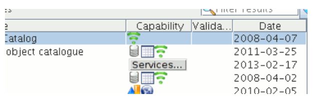
The registred data:

```
<ri:Resource [...] xsi:type="vs1:DataCollection">
  <title>Apache Point observations of lensed quasars</title>
  [...] <content> [...]
    <relationship>
      <relationshipType>served-by</relationshipType>
      <relatedResource
        ivo-id="ivo://org.gavo.dc/lensunion/q/im"
        >Lens Image Archive</relatedResource>
      <relatedResource
        ivo-id="ivo://org.gavo.dc/__system__/tap/run"
          >GAVO Data Center TAP service</relatedResource>
    </relationship>
  </content>
```

Note how the data is exposed both using a „custom“ SIAP/web service and throught ObsTAP.

5. ...in VOResource, service

```
<ri:Resource [...] xsi:type="vs:CatalogService">
  <title>Lens Image Archive</title>[...]
    <content>[...]
      <relationship>
        <relationshipType>service-for</relationshipType>
        <relatedResource
          ivo-id="ivo://org.gavo.dc/apo/res/apo/frames"
            >Apache Point observations of lensed quasars
        </relatedResource>
        <relatedResource
          ivo-id="ivo://org.gavo.dc/danish/red/data"
          >Danish Observatory Lens Images</relatedResource>
        [...]
      </relationship>
    </content>
```



e	Capability	Valida...	Date
Catalog			2008-04-07
object catalogue			2011-03-25
	Services...		2013-02-17
			2008-04-02
			2010-02-05

Fig. 1

6. The Server Operator's Perspective

With the publishing suite DaCHS[1], registering a TAP-accessible table is as easy as saying:

```
  <table onDisk="True" id="objects" adql="True">
    <meta name="title">My objects table</meta>
    <register/>
    ...
```

A table being served through some SIAP service would say:

```
  <table id="rawframes" adql="True" onDisk="True">
    <register services="lensunion/q#im"/>
    ...
```

Admittedly this is that easy because all the metadata already is defined somewhere in the files these snippets come from, and further heavy lifting – the delivery of the resulting resource records into the VO registry – is done through the OAI-PMH interface built into the software. Still, once a publishing toolkit has enough metadata for registration itself, the additional effort is very low indeed.

Forms-based registry record management: At least for TAP, we could help people by grabbing data from VOSI.

7. The user perspective

Registry clients must show records for data collections much like they show those for services; the information is there – here's how to figure out access URLs for a data item in our current relational registry plan:

```
SELECT url FROM rr.relationship AS a
  JOIN rr.accessurl AS b ON (related_id=b.ivoid)
WHERE
  a.ivoid='ivo://org.gavo.dc/gums/q/pub'
  AND relationship_type='served-by';
```

I want of VODesktop back . . .

For an interface leading from data to services, I see the following options for registry UI authors:

- Display all relationships in some way (this would include things like derived-from or related-to)
- Treat served-by and service-for in a special way.

I'd always vote for the second option, since at least served-by has high functional implications. Most other relations are more provenance-like and do not really determine what people can "do" with the resource. Mirror-of is a bit in-between: I can see why a UI might want to offer immediate access to mirrors, too.

Sure, that interface shouldn't be an ugly button. But I do see a popup here, since none of the relationships are 1:1.

(vgl. Fig. 1)

[1] http://soft.g-vo.org/dachs