1. A Praise of Datalink

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

TL;WL: Datalink lets you decouple request from dataset from what’s on your disks in ways that make many things that have been painful with DaCHS in the past sheer bliss.

2. Where’s Datalink?

So far: Data discovery lead directly to a dataset (FITS image, VOTable, whatever)
Now: it can also yield a VOTable subtype identified with application/x-votable+xml;content=datalink

The access URL points to a service accepting a single ID parameter.

3. Datalink documents

A datalink resource contains a table of access options. Here’s an artist’s impression of one:

<table>
<thead>
<tr>
<th>ID</th>
<th>Acc.URL</th>
<th>Svc</th>
<th>Description</th>
<th>Sem</th>
<th>MIME</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>loc://a/b</td>
<td>s.org/id</td>
<td></td>
<td>High res</td>
<td>data</td>
<td>link</td>
<td></td>
</tr>
<tr>
<td>loc://a/b</td>
<td>s.org/lsap</td>
<td></td>
<td>The pixel in a db table</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>loc://a/b</td>
<td>s.org/gst</td>
<td>@def</td>
<td>Data access</td>
<td>access</td>
<td></td>
<td></td>
</tr>
<tr>
<td>loc://a/b</td>
<td>s.org/get?url=0-True</td>
<td>Full DS</td>
<td>self</td>
<td>fits</td>
<td>S69</td>
<td></td>
</tr>
</tbody>
</table>

If you’re wondering what the purpose of the constant ID column is: It is possible to ask for datalinks for multiple datasets. Responses for different datasets would be told apart using this column.
See this¹ for a live example.

4. Datalink Service Descriptions

The #sdef in the last example reference something like:

```xml
<RESOURCE ID="ahuwdnd" type="service">
  <GROUP utype="stc:CatalogEntryLocation">
    <PARAM utype="stc:...CoordFlavor" value="SPHERICAL"/>
    <PARAM ref="ahwloee" utype="stc:HiLimit2Vec.C1"/>
    <GROUP name="inputParams">
      <PARAM name="ID" ucd="meta.id;meta.main">
        <DESCRIPTION>The publisher DID of the dataset of interest</DESCRIPTION>
      </PARAM>
      <PARAM name="DEC_MIN" ucd="par.min;pos.eq.dec" unit="deg">
        <VALUES>
          <MIN value="260.934294375"/>
          <MAX value="260.958193719"/>
        </VALUES>
      </PARAM>
      ... (more parameters) ...
      <PARAM name="accessURL" ucd="meta.ref.url" value="http://dc.zah.uni-heidelberg.de/califa/q/dl/dlget"/>
    </GROUP>
  </GROUP>
</RESOURCE>
```

What does all of this mean? A service description to a first approximation is a sequence of input parameters, each coming with UCD, unit, and most importantly ranges of valid values.
In addition to these, we have an STC group so clients can figure out details like coordinate systems and such.
In this example, clients can also do cutouts using pixel coordinates.

¹ http://dc.zah.uni-heidelberg.de/califa/q/d1/dmeta?ID=ivo%3A//org.gavo.de/%7E%3Fcalifa/data/V500/reduced_v
5. Data Access Parameters

I strongly advocate atomic PARAMs with rich metadata (ranges, enumeration of valid values). Plus, there should be reserved name/ucd/unit combinations for certain types of physics:

<table>
<thead>
<tr>
<th>Name</th>
<th>UCD</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>meta.id;meta.main</td>
<td>(none)</td>
</tr>
<tr>
<td>DEC_MIN</td>
<td>param.min;pos.eq.dec</td>
<td>deg</td>
</tr>
<tr>
<td>DEC_MAX</td>
<td>param.max;pos.eq.dec</td>
<td>deg</td>
</tr>
<tr>
<td>RA*</td>
<td>param.*;pos.eq.ra</td>
<td>deg</td>
</tr>
<tr>
<td>LAT*</td>
<td>param.*;special</td>
<td>deg</td>
</tr>
<tr>
<td>LON*</td>
<td>param.*;special</td>
<td>deg</td>
</tr>
<tr>
<td>LAMBDA*</td>
<td>param.*;em.wl</td>
<td>m</td>
</tr>
<tr>
<td>FORMAT</td>
<td>meta.code.mime</td>
<td>(none)</td>
</tr>
<tr>
<td>SPECRP*</td>
<td>param.*;spect.resolution</td>
<td>(empty)</td>
</tr>
<tr>
<td>FLUXCALIB</td>
<td>phot.calib</td>
<td>(none)</td>
</tr>
<tr>
<td>PIX(n)*</td>
<td>param.*;pos.pixel.ax(n)</td>
<td>pixel</td>
</tr>
<tr>
<td>KIND</td>
<td>meta.code</td>
<td>(none)</td>
</tr>
</tbody>
</table>

As you can see, I’m suggesting some additional UCDs. I’m not convinced the “param.*” actually catches the semantics. It should be something like a “generic limit”; “stat.max” pretty certainly doesn’t fit here.

Incidentally, there’s something to be said for choosing VAL/SIZE instead of MIN/MAX. It’s more robust against weird spots in (spherical, in particular) coordinate systems, for starters. It’s also a bit larger burden on the server implementation, of course.

Plus, we’ll need more experience if clients can actually work out the interfaces from this. From the SPLAT experience, we’d say they mostly can, but more evidence is needed.

6. DAL associate services

In S*AP or ObsTAP service responses:

```xml
<GROUP utype="stc:CatalogEntryLocation"> ...
</GROUP>

<PARAM name="ID" ref="ssa_pubDID" ucd="meta.id;meta.main"/>
<PARAM name="FLUXCALIB" ucd="phot.calib" utype="ssa:Char.FluxAxis.Calibration">
  <DESCRIPTION>Recalibrate the spectrum....</DESCRIPTION>
  <VALUES>
    <OPTION value="RELATIVE"/>
    <OPTION value="UNCALIBRATED"/>
  </VALUES>
</PARAM>
<PARAM name="LAMBDA_MIN" ucd="par.min;em.wl" unit="m"/>
<PARAM name="accessURL" ucd="meta.ref.url" value="http://localhost:8080/flashheros/q/sdl/dlget"/>
</GROUP>

To have this point to a datalink service:

```xml
<PARAM name="standardID" datatyp="char" arSmarty="*" value="ivo://ivoa.net/std/Datalink#File"/>
```

7. Boring Cube Case

Slicing and dicing cubes. Discovery via obscore.

- Link to high/low-resolution cubes among themselves
- Link to a TAP service that contains all the voxels
- Link to the full cube
- Link to a service with parameters RA*, DEC*, LAMBDA*, PIX*, KIND, including STC metadata definition.

... all this in about 60 lines of resource descriptor.

live; also, live for a slit spectrum4, also note the link to a separate errors file here.

8. Boring Son of SSA getData

This is about cutouts (LAMBDA*), recalibration (FLUXCALIB), and formatting of spectra (FORMAT); it’s particularly nice if you don’t want to touch spectra in 1D images and yet deliver standards-compliant spectra.

live on CALIFA3, live on FEROS5, live on Flash/Heros6, live on TheoSSA7 (look for the RESOURCE type='service'),

... all this in 20 to 50 lines of RD (depending on how much code is needed to get to the lambda/flux pairs).

9. Spectra from the DB

GAIA simulated spectra came in GBIN archives (shudder). To serve them, write a datalink service pulling the spectral points out of a postgres array.

50 lines of RD each for two different cases (RP/BP, RV), and access URLs go directly into the datalink service.

RP/BP live8, RV live9.

---

2 http://dc.g-vo.org/califa/q/kl/dlmeta?ID=ivo%3A%2F%2Forg.gavo.dc%2F%7E%3Fcalifa%2Fdata%2FV500%2FReduced_v1
3 http://localhost:8080/sionalgo/g/kl/dmeta?ID=ivo%3A%2F%2Forg.gavo.dc%2F%7E%3Fcalifa%2Fdata%2FV500%2FReduced_v1
4 http://dc.g-vo.org/califa/q/ssap.xml?REQUEST=queryData
5 http://dc.g-vo.org/feros/q/ssap.xml?REQUEST=queryData
6 http://dc.g-vo.org/flashheros/q/ssap.xml?REQUEST=queryData
7 http://dc.g-vo.org/theossa/q/ssap.xml?REQUEST=queryData
8 http://localhost:8080/c8spect/q/rpbps/ssap.xml?REQUEST=queryData
9 http://localhost:8080/c8spect/q/rvs/ssap.xml?REQUEST=queryData
10. Quick Echelle Prototype

Wanted to publish split-order Echelle spectra. No need to change the server software, wrote about 150 lines of RD. Ok, this is cheating, I did write more code and added it in the server to parse the extra MIDAS tables necessary to understand the input format, but that doesn’t really count.

Again SSA accrefs go directly into datalink.

live^{10}

11. Hairs in the Soup

Gripes with the current draft:

* The REQUEST parameter should die
* RESPONSEFORMAT prose should go
* We really, really, really should have the conventional parameters for data access services in the WD. Really.

^{10} http://dc.g-vo.org/flashheros/q/echssa/ssap.xml?REQUEST=queryData