Fig. 1

Fig. 2


Fig. 3

1. A Praise of Datalink

(cf. Fig. 1)

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(cf. Fig. 2)

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.

(cf. Fig. 3)

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`

```
tapsh> select top 1 access_format, access_url \
from ivoa.obscore \
where access_format like '%datalink'!

access_format            access_url
[...];content=datalink   http[...]?ID=[...]NGC6310.V500.rscube.fits
```

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:

ID	Acc.URL	Svc	Description	Sem	Mime	Len
ivo://a/b	s.org/dl		High res		datalink	
ivo://a/b	s.org/tap		The pixels in a db table			
ivo://a/b	s.org/get	#sdef	Data access	access		
ivo://a/b	s.org/get?full=True		Full DS	self	fits	5e9

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 this[1] for a live example.

4. Datalink Service Descriptions

The `#sdef` in the last example reference something like:

```
<RESOURCE ID="ahuwdnd" type="service">
  <GROUP utype="stc:CatalogEntryLocation">
    <PARAM utype="stc:...CoordFlavor" value="SPHERICAL"/>
    <PARAMref ref="ahswloee" utype="stc:HiLimit2Vec.C1"/>
  <GROUP name="inputParams">
    <PARAM name="ID" ucd="meta.id;meta.main">
      <DESCRIPTION>The pubisher DID of the dataset of interest
    <PARAM name="DEC_MIN" ucd="par.min;pos.eq.dec" unit="deg">
    <PARAM name="DEC_MAX" ucd="par.max;pos.eq.dec" unit="deg">
    <PARAM name="RA_MIN" ucd="par.min;pos.eq.ra" unit="deg">
    <PARAM ID="ahswloee" datatype="float" name="RA_MAX"
       ucd="par.max;pos.eq.ra" unit="deg" value="">
      <DESCRIPTION>The longitude coordinate, upper limit</DESCRIPTION>
      <VALUES>
        <MIN value="260.934294375"/><MAX value="260.958193719"/>
    <PARAM name="COO_3_MIN"><PARAM name="COO_3_MAX">
    <PARAM name="PIXEL_1_MIN" ucd="par.min;pos.cartesian;instr.pixel">
  <PARAM name="accessURL" ucd="meta.ref.url"
    value="http://dc.zah.uni-heidelberg.de/califa/q/dl/dlget"/>
```

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/dl/dlmeta?ID=ivo%3A//org.gavo.dc/%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:

Name	UCD	Unit
ID	meta.id;meta.main	(none)
DEC_MIN	param.min;pos.eq.dec	deg
DEC_MAX	param.max;pos.eq.dec	deg
RA_*	param.*;pos.eq.ra	deg
LAT_*	param.*;(special)	deg
LON_*	param.*;(special)	deg
LAMBDA_*	param.*;em.wl	m
FORMAT	meta.code.mime	(none)
SPECRP_*	param.*;spect.resolution	(empty)
FLUXCALIB	phot.calib	(none)
PIX_(n)_*	param.*;pos.pixel.ax(n)	pixel
KIND	meta.code	(none)

Your contribution here

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 cliens 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:

```
<RESOURCE ID="lghhliw" type="service">
  <GROUP utype="stc:CatalogEntryLocation">...
  <GROUP name="inputParams">
    <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....
      <VALUES>
        <OPTION value="RELATIVE"/><OPTION value="UNCALIBRATED"/>
   <PARAM name="LAMBDA_MIN" ucd="par.min;em.wl" unit="m"
 </GROUP>
 <PARAM name="accessURL" ucd="meta.ref.url"
  value="http://localhost:8080/flashheros/q/sdl/dlget"/>
```

To have this point to a datalink service:

```
<PARAM name="standardID" datatype="char" arraysize="*"
  value="ivo://ivoa.net/std/DataLink#links" />
```


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_n_*, KIND, including STC metadata definition.

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

live[2]; also, live for a slit spectrum[3], 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 CALIFA[4], live on FEROS[5], live on Flash/Heros[6], live on TheoSSA[7] (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 live[8], RV live[9].

[2] http://dc.g-vo.org/califa/q/dl/dlmeta?ID=ivo%3A%2F%2Forg.gavo.dc%2F%7E%3Fcalifa%2Fdata%2FV500%2Freduced_v1
[3] http://localhost:8080/mlqso/q/d/dlmeta?ID=ivo%3A%2F%2Forg.gavo.dc%2F%7E%3Fmlqso%2Fdata%2Fslits%2FQ0142_dat
[4] http://dc.g-vo.org/califa/q/s/ssap.xml?REQUEST=queryData
[5] http://dc.g-vo.org/feros/q/ssa/ssap.xml?REQUEST=queryData
[6] http://dc.g-vo.org/flashheros/q/ssa/ssap.xml?REQUEST=queryData
[7] http://dc.g-vo.org/theossa/q/ssa/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`