Fig. 1

Fig. 2

1. Flexible accessData using Datalink

(cf. Fig. 1)

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(cf. Fig. 2)

This is shameless advertising for using a simple, proven method to implement "accessData" (AD).

TL;DR: All metadata unambiguously declared in a VOTable. Design from the client towards the server. Don't unnecessarily restrict scope.

2. Principles

- AD services are normal datalink services (i.e., metadata in VOTable params plus model annotation)
- Parameter types are what's given by VOTable, strings+syntax discouraged
- Asymptotically total metadata definition
- Operators are free to expose whatever parameters are useful for their data
- Standard parameters are extension-safely defined through three-factor semantics
- Deployments now should not *need* to be touched on updates

3. Three-factor semantics

In order to allow clients to naively use simple names for doing cutouts (or whatever) on well-known quantities, there are reserved name/ucd/unit combinations, e.g.:

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

A client only interested in doing cutouts in ICRS would simply check if the three factors in the metadata match whatever is in its list of standard semantics and discard the service if they don't. It can use plain key-value pairs afterwards with impunity otherwise. The three factors ensure that names of today's custom parameters can be used in tomorrow's standards without clashes.

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.

We'll need a bit 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.

4. What's so great about it?

- No ad-hoc syntax (in the standard)
- Rich interfaces (including limits, pre-filled fields) easily implementable even for unknown physics
- Model-neutral (but standard DM annotation possible as DMs become available)
- Straightforward discovery and use of known physics
- Minimal spec effort (could be a chapter of datalink)
- Existing integration of STC with a clear evolution perspective
- Proven record on client *and* server of a similar technique in SSAP/Splat

5. Counterarguments

- Async services (this *probably* is just a UWS application, but since I've not implemented that yet, I won't make promises)
- Unwieldy with funny coordinate systems (e.g., poles of spherical coordinates): VAL/SIZE instead of MIN/MAX?
- You tell me...

6. Services doing this

In case you want to see metadata for datasets:

- CALIFA cube[1]
- 2D slit spectrum[2]
- spectrum with cutout, recalibration, and such[3]
- completely virtual spectrum[4]
- Accessing individual Echelle orders[5]

Set up your own service using DaCHS: Docs[6] Think that's ugly? Well:

Clients, clients, clients!

[1] http://dc.g-vo.org/califa/q/dl/dlmeta?ID=ivo%3A%2F%2Forg.gavo.dc%2F%7E%3Fcalifa%2Fdata%2FV500%2Freduce

[2] http://dc.g-vo.org/mlqso/q/d/dlmeta?ID=ivo%3A%2F%2Forg.gavo.dc%2F%7E%3Fmlqso%2Fdata%2Fslits%2FQ0142_da

[3] http://dc.zah.uni-heidelberg.de/flashheros/q/sdl/dlmeta?ID=ivo://org.gavo.dc/~?flashheros/data/ls95/blue/f1938.mt

[4] http://dc.zah.uni-heidelberg.de/c8spect/q/rvdl/dlmeta?ID=ivo://org.gavo.dc/~?c8spect/rpbp/1152923226888832682

[5] http://dc.zah.uni-heidelberg.de/flashheros/q/echdl/dlmeta?ID=ivo://org.gavo.dc/~?flashheros/data_raw/ls97/blue/n1544.mt

[6] http://docs.g-vo.org/DaCHS/ref.html#datalink-cores