Fig. 1

Fig. 2

1. What is VO Publishing?

(vgl. Fig. 1)

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(vgl. Fig. 2)

- Level -1: Web, Data Formats
- Level 0: Registry
- Level 1: Data Models
- Level 2: Protocols
- Checks, Balances, and all that.

For the initiated: These "levels" are ad-hoc for this talk. They are inspired by, but not identical to, VO resource metadata validation levels.

2. Publishing, Level –1

Publishing means offering data to other people. So

1. there must be some way for them to get it
2. they must have a fair chance to recover actual values from the bytes they get

For (1), the Web (i.e., HTTP-based browser services) is a fair *start* these days.

For (2), there's data formats. The VO likes VOTables, FITS, VOEvent. But if you do it right, CSV or HDF5 might count, too.

3. Level 0: Registry

To let people find your service – from within applications, via the web, or whatever – and reference it properly, you need to get your service's metadata into the VO ("register it"), and you'll get back an identifier back, an IVORN looking like ivo://authority/local/part.

You can and should register browser-based services, even data collections residing on FTP servers.

Getting into the registry is not terribly hard. It basically means that you publish structured metadata ("resource records") about your service – title, description, creator, maintainer, keywords, access options, and the like. Within the Registry (upper case: the abstract concept "collection of all resource records"), the metadata for one service is identified by a URI, the IVORN.

You do *not* need to run your own publishing registry to have your resources in the registry.

You may (but need not) create an authority of your own; your service could then be called `ivo://my.shiny.obs/astounding`!

4. Level 1: Data Models

A format like FITS or VOTable lets you say

- here's a bunch of tables,
- the columns of which have these names, types, units, and descriptions,
- and this is how you get the values

A data model lets you, in addition, say things like

- this is a spectrum,
- obtained on A at B by C using D,
- pointing at ICRS coordinates α, δ
- with fluxes over wavelength there,
- and flux errors there;
- the spectral range is X through Y
- and so on, on, and on.

The VO defines data models for spectra, characterization, coordinates, events, core observation data, and many other things.

Make your data compliant to data models, and clients (i.e., user programs) can process your data with much less user intervention (and/or guessing); – this usually is a tradeoff of desirable thoroughness versus getting the data out; the optimal stopping point is hard to determine, in particular without divination powers.

5. Level 2: Protocols

Protocols let programs ("clients") talk to your server in a well-defined fashion.

This is just as you can access an HTTP server using firefox, curl, the python standard library, etc. Most VO protocols are built on top of HTTP and specify what parameters come in, the data model of the response, how to signal errors, etc.

In the VO, you can publish arbitrary tabular data via TAP. This is a bit like the protocol equivalent of a generic data container format: very flexible, but without sufficient semantics for a single case. To give tables semantics, use data models.

Obscore via TAP lets you publish arbitrary observations-like data. But note that client support is not amazing so far; so, if you can, use one of the typed protocols, possibly in addition to ObsTAP:

Simple Cone Search (SCS) is for publishing catalog-like data.

Simple Image Access (SIAP) is for publishing astrometrically calibrated image-like data

Simple Spectral Access (SSAP) is for publishing spectra (and to some degree time series)

For all the typed protocols, there is reasonable client support (e.g., in Aladin, TOPCAT, Splat, VOSpec, and so on). This means you can run queries like "give me all spectra in the VO for objects 0.5 arcminutes around M51 taken in November 2008" or so.'

6. Checks and Balances

Being part of the VO also means being part of a vast distributed information system.

It can tolerate a bit of lint, but bad implementations or rotting installations cause bad user experience. Therefore:

Good VO citizens validate their services

Use validators; just because your service seems to work fine the one program, it doesn't mean it's standards compliant.

See

http://wiki.ivoa.net/twiki/bin/view/IVOA/ServiceValidation

for links to validation software.

7. In Conclusion

Publishing to the VO means, in rough dependency order:

1. Registering what one has
2. Fixing up one's files to make them rich in metadata and standards-compliant
3. Making one's server speak the appropriate protocol
4. Validating the whole thing
5. Feeling really good for having made a contribution to humanity's betterment