Coping with Major Version Changes

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

e-inf-astro

Starting point: Add `servicetype='sia2'` to pyvo's registry.search.

So far, `'image'` was an alias for `'sia'`.

What should it be now? More generally: What's sane client behaviour on major version changes?

Opening Remark

Don't use servicetype constraints going forward.

Data product types are not a property of the service protocol (think obscore, think timeseries in ssa). They are a property of the data collection (which can be accessible through multiple protocols).

Let's further move to *data discovery* and add a product-type declaration to VODataService

Drop image?

Option I: `servicetype='image'` will give a DeprecationWarning and keep current behavoiur otherwise.

If we are serious about getting rid of servicetype as fast as possible, that's the way to go. It's also the simplest options implementationally.

Break image?

Option II: Make `servicetype='image'` equivalent to `servicetype='sia2'`.

This would be right if we expected SIAP2 to take over all of SIAP. Eight years after SIAP2: 271 SIAP1 vs. 99 SIAP2. Also: SIAP2 services have a rather different interface from SIAP1 services.

Fix image?

Option III: Make `servicetype='image'` return some sort of "union" of sia and sia2.

Problem: What interface would that have? A program has know what parameters it can pass to a service object.

The only solution I can see is to return adapter objects that support POS and returns obscore-like records for both sia and sia2.

De-duplicate?

But many resources have both sia and sia2 capabilities.

- Option IIIa: produce service objects for both
- Option IIIb: prefer SIAP1
- Option IIIc: prefer SIAP2

And Obscore

Of course, there's also Obstap that people can use to discover images. Why shouldn't that come back for the servicetype image?

...ignoring for a moment the difficulty of figuring out the pertinent value of obs_collection.

In the End

What's your preference for limping on with service discovery?

But I'd still say: Let's move to data discovery and have per-resource decisions what protocol to use.