# IVOA Nov 2021 Interop - DAL / SSIG

Tuesday 2 November 2021 - 13:30 UTC - vconf

Participants: (58) James Dempsey, Grégory Mantelet, Marco Molinaro, Mark Cresitello-Dittmar, Laurent Michel, Pierre Fernique, Daniel Durand, Markus Demleitner, Brian McLean, Mickael Boiziot, Patrick Dowler, Marianne Brouty, Mark Taylor, Tess Jaffe, Yannick Roehlly, Jean-Michel Glorian, Anne Raugh, Brian McLean, Janet Evans, Ada Nebot, Franck Le Petit, Cyril Chauvin, François Bonnarel, Christophe Arviset, Jesus Salgado, Kai Polsterer, Margarida Castro Neves, Trey Roby, Adrian Damian, Anastasia Galkin, André Schaaff, .....

## Agenda:

1. Availability after Caproles
2. EPN-TAP v2.0 RFC
3. ADQL v2.1 Validation

## Availability after Caproles

* The purpose of VOSI a10y is to tell whether a service is responsive
* Optionally, it is possible to say since when a service is down or up
* VOSI requires one availability per service, but could have TAP, SCS capabilities which might have seperate uptime, also mirrors might also have varying uptime
* 3 proposed solutions:
    1. Forget about availability ; currently there is no interoperable way to use this endpoint. Generally a client tries to use the service and generally not test it using the availability endpoint
    2. Per-interface endpoints : declare explicitly the URI/URL of the availability endpoint of each underlying service
    3. Multi-endpoint response : group the availability of each underlying service at the top level in the <availability> node: one <endpoint> element in <availability> ; so we keep one availability per-resource

Pierre: suggestion to have multiple accessUrls

Tom Donaldson: if services are on the same service their down time is generally the same and in the case they are on different machines it requires a lot of monitoring

Pat Dowler : the code monitoring the service is inside the service's code. It checks things like connection to the database. This kind of availability maintenance is useful for operators to know the monitore the service availability and useful for the users to know when they can try accessing the service. S3 less interesting. Then S2. Assume S1 would be the preferable solution.

Trey Roby: Wonder if it would be premature to forget about availability if in the future technologies like Kubernetes helps to maintain reliable availability endpoints.

FB: we could have availability optional. Capability would tell you if it's there or not !
MD: We'd still have to fix it, and from what I see now it doesn't seem there's sufficient demand for putting in the work required.
MM: can you expand on "fix it"? If we go S1, with "may" have availability in DALI and leaving VOSI as is, wouldn't it work? (I was checking the RECs to figure this out)
MD: No, because availability is a property of the access/mirrorURLs, not of the service/resource.  Hence, we'd have to adopt something like S2 or S3 anyway (or invent some horrible hack for how to associate accessURLs within the a10y capability with those everywhere else -- ugh).
MM: OK, think I understand it. So, an optional S2 could be a solution, but an optional S1 is not.
MD: Roughly, yes.  "Optional S1" is what we have now in practice, because people chose to ignore our requirements for a10y capabilities...

# EPN-TAP v2.0 RFC

* EPN-TAP : Usual TAP mechanism with EPNCore metadata vocab
* Status:

- * PR issued at late July 2021 after large doc update
- * RFC open since end of September until 14 Nov.

* Validation implemented in TAPLint
* Implementations:

- * 57 service public EPN-TAP services in registry, 20+ have been updated
- * EPN-TAP in DACHS-beta: existing mixins, tutorial
- * Also 2 implementations based on VOLLT - ESA - 20M rows and at CDS for Vizier - see poster
    - * GL: VizieR catalogue (curated by VO Paris) https://vizier.cds.unistra.fr/viz-bin/VizieR?-source=B/planets
    - * SE: Same one seen in the VESPA portal: http://vespa.obspm.fr/planetary/data/display/?&service_id=ivo://cds.vizier/b/planets&service_type=epn

In future:
* new parameters
* requests for new UCDS expected
* need to access datalinks for several granules

- MD: Oh yes, you can have multi-ID requests even now. Just repeat the ID parameter.

* flexibility expected in ADQL (e.g. pagination)
* extra standards required (for target names, coordinates systems, obs/space missions)

* Existing services are in v2.0 (updated to last version)
* EPN-TAP services entries to be reviewed in the registry (many remnant of older versions)
* XSD schema to update

FBonnarel: I think it's possible to access several #this in the same datalink query, using the multi-ID feature? Is that answering Stephane's concern ?
Ah I agree with Markus (see above)

Last version is dated 22 Oct but copy on documents page is older so might be best to review GitHub copy

# ADQL v2.1 Validation

- Ready for RFC but need validation
- Problem - No parsers for the BNF which support testing queries against the BNF
- One approach was lyonetia but the BNF validation was not successful (MD: of course, the main idea was to give ADQL implementors a set of queries to test their implementation against...).
- Two contributors to Lyonetia only - GAVO and Edinburgh
- New proposal
    - use a parser not related to a specific service - VOLTT/ADQL-Lib
    - Validate queries but don't run them
    - Use XML format defined for Lyonetia by Dave Morris
- Progress
    - Java based - jar file or can be run with gradle
    - Can validate individual files or a directory
    - Aiming to run in GitHub/CI and report in markdown
- Showed example text and markdown reports
- Validator almost complete
- Need volunteers to provide ADQL queries for testing
- RFC could be started in parallel once validator completed and public

LM: Working on a query tool and could generate some complex queries for validation
DM: Really good work and hard to get off the ground
DM: Is it possible to get a second parser to process the test queries?
MD: Could validate the PEG grammar

DM: PEG not in the standard yet, ADQL 2.1 based on BNF still
GM: Would still be valid to test based on the PEG
MD: Could also use DACHS to parse there - would test that a human can
MD: Examples need to be artificial - create query for each rule in the standard (much like unit tests) - would volunteer to do ~10 rules
GM: First query from DaCHS with nulls was difficult - MD: Should be removed - not necessary.
DM: Edinburgh can contribute some test queries for specific rules
JD: CASDA can also do some test queries