Protocol Transitioning Tiger Team (P3T)

March Meeting - 26 Mar 2024 20:00 UTC

Attendees: James Dempsey, Paul Harrison, Marco Molinaro, Joshua Fraustro, Janet Evans, Russ Allbery, Tom Donaldson, Jesus Salgado, Gregory Mantelet, TS dower, Gregory Dubois-Felsmann, Dave Morris, Sara Betocco

Actions from Previous Meeting

- RA: email out ideas on error handling based in part on rubin work.
- DM: work on spring server implementation from Russ' cone search OpenAPI spec
- JD: Assist with spring implementation
- RA: Explore rust implementation against OpenAPI if possible
- MM: check out other frameworks and what is out there

Structured Error Handling

HATEOAS (https://en.wikipedia.org/wiki/HATEOAS) - prior ART for JSON/REST - used by GitHub - links to a page for each error with some descriptions
We need to provide a way for sites to have their own errors, either site-specific codes or URLs ; feedback from Dave M - Russ and Dave in sync.
JD: Only concern is effort to implement the codes and pages on providers
RA: Could make these feedback mechanisms optional and provider builds up as they evolve
TD: Agree, making as many things optional as possible are fine if user experience related; few options with programatic access to these things. Russ agrees.
PH: Unifying error messages needs to be considered; not in individual standards (e.g., VOTable)
GD-F: Very intersted in seeing a well established error systems, for ADQL as the messages are about the content of their query
JF: Not sure we would send back an HTTP error code for an ADQL syntax error - more likely in table response.
DM: ADQL a good example; sync vs async - sync can provide an immediate response, but async might need to accept a few parameters and then parse/validate it at runtime - so http error codes might not be appropriate for async
GD-F: Many different implementations in the wild
RA: Hard to distinguish error in a cutout example; argue to make sure whether the result is an error or success; indicate error with http code (there are differing opinions) - keep layers separate. For async UWS doesn't have a way to indicate a

job has partly succeeded and party failed. Want errors in one place and success in another. Not mixed.
JD: Agree, had similar experience.
JF: For UWS, Async requests - allow no-ADQL; Parse twice ... when job run it's valid -> send to parser to translate to DB query, and then another check and return of not correct.

GD-F: Stack of issues - UWS allows jobs to be editted up until it is submitted. Potential convenience to live user but not programmatic. e.g. upload might happen after the ADQL is provided. Agree it is confusing but there is a purpose.

GD_F: On sync - agree 200 response for error is undesirable. Would be better to provide a 4xx rewpsonse to avoid the caller needing to search for an error message. For async, multi-results case do get complex as parameters combine into an outer product of the criteria. Difficult to map resiults back to inputs.

MM: 4xx error codes are alreayd in DALI section 4, some legacy protoicols have votable still.

MM: Multi-cutout - dividing sync and async defined the same - should we impose restrictions such as sync such as can only produce single cutout, must use async if want multiple cutouts. VOTables can communicate non breaking errors (e.g. overflow).

PH: Historically people wanted to seperate transport from protocol, hence resistance to having http error cod

TD: Need to define a migration path for protocols atht currently use votables for errors.

OPEN API

JD: 3 ppl tasked with looking at OPEN api spec: Dave, James, Russ
Dave: looking at open apo - haven't sompleted actions - cone search spec and develop .
JD: straight forward to prodduce service stubs; stubs for each api calls; will use the models and APIs as reference when implementating
RA: Used Rust; 3.1.0 a major change, but a good one; community hasn't settled yet;; non-trivial change ... generators not up to speed' couldn't find one in Rust. Didn't get to complete task before meeting given the mismatch and tedious work to get them in sync invovled.

openAPI 3.1.0 includes support for json schema which is a big change. Suggest we target 3.0.2 for the schema. Not difficult but tedious.

Other frameworks

MM: looked at GraphQL - a way of using JSON to provide general interfaces; missing functionalities; cannot see path with ADQL - Not recommended
RA: 2nd that thought - not a good fit without thinking about replacing ADQL; Assumes your world is documentas represented in JSON - not what we have. Whole language - need to go all in.
TD: Lower level accesses to ST archive queries are now being done with GraphQL; get executed on SQL DBs with an option to change the backend (e.g., Cloud, NoSQL). Used for basic geometries / regions. Could be thought about in the future.

PH: https://www.oxlip-lang.org/doc/related.html; other approaches that are quite good ... this is just an example.
PH: Slight issue with OPENAPI is modularity; will have a lot of repeated stuff - can use inclusions, but that is better specified in 3.1
JD: OPEN API takes in YAML and that was good.

What comes next - What do we produce from the Sydney mtg

JD: UWS spec - how to harmonize those to evaluate the transition cost - UWS & Russ's spec??
PH: Think JF's was cloaest to the original spec
JF: Stayed close but the binder one may tbe the one to go after.
GD-F: Purpose of minimizing changes?? Write an entire front end service over an existing backend; we also talked about writing adapters at one point. Is it still an aspiration?? Argument: many more clients than services; clients understand the old and new way; servies would be able to just switch and not worry about maintaining 2 implementations.
JD: Agree in TAP/ADQL ; ConeSearch has a lot of clients - need to tread carefully
GD-F: Cone Search is multiple generations of the standards behind, of course.
I think we just have to recognize that many of the older cone search services are ~never going to go away.
... and therefore the associated clients will have to be preserved as well.

RA: Overall question about making the protocol better in the future; nice to figure out a way that the problem becomes more contained; have that picture to modularize so that future protocol transitions are easier. Meta architecture on how protocols in IVOA could be written to support that change. DALI and UWS apply across multiple protocols - that sort of layering is a good thing. Providers register both old and new protocols and serve both. Client can request json or xml so client can indicate which protocol it wants. But might still be easier to have old and new on different URLs than use content negotiation.
MM: Agree; like the idea of content negotiation between json and url; Idea of known relational things behind the scenes - do we need to think about that in this transition?
G D-F: Mario presentation in Bologna - HIPScap (parquet spced catalogs in hopefully standard way) - not a relational DB underneath, SQL adapter strategies to connect it to a SQL query; sometime limitations in this configuration. Could

think in this direction instead of trying to shoe horn into ADQL. We're doing this for new services. DALI 2.0 could be an outcome of this work - query service APIs from these discussions.
TD: Agree with G D-F. Could come up with new ways to define things ... standardization of new languages/query types could be separate.

Efforts over the next month
RA: 1) Effort toward structure error message API; 2) Raise thousght on structure for new protocols; use case that G D-F raised; services for Rubin that follow a pattern but don't raise to level of IVOA - still commality.
JD: Exploiting OPEN API further
DM: OPEN API for execution planner proposal - ready for May
ALL: Think about format of the 2 day meeting in Sydney and the agenda for that meeting

References

- https://github.com/spacetelescope/vo-openapi/blob/main/openapi/uws/uws.yml - YAML OpenAPI model for UWS by Joshua Frausto
- https://github.com/ivoa/PTTT/tree/main/cone-search - OpenAPI model for COneSearch and a hypothetical UWS implementation for ConeSearch by Russ Allbery
- https://rds.org.au/events/ivoa-2024/ - Meeting page for the May 2024 Interop in Sydney
- https://wiki.ivoa.net/twiki/bin/view/IVOA/InterOpMay2024 - Draft scheduled for the May 2024 Interop in Sydney

Actions

Next meeting

Mon 22 Apr 20:00 UTC