Friday 17 May 2024 - P3T day two

Attendees: In person -
Joshua F, Dave M, Tom D, Marco M, James D, Pat D, Gregory M, Jesus S, Janet E;
Online: Sara B, Gregory DF

Draft agenda

- 10:00 - 10:15 - Gathering
- 10:15 - 11:15 - Session 5
    - Recap of outcomes from yesterday's session
    - Talks for the interop session - split into smaller groups to work on each of the four talks
        - Introduction to the protocol transition project -> Janet
        - Technical changes overview -> Joshua
        - Implementation approach -> Gregory DB
        - Implementor responses -> James
    - Determine who will be liaising with the authors of each of the major clients
- 11:15 - 11:30 – short break
- 11:30 - 12:30 - Session 6
    - Discuss talks and work on these further
    - Outline of documents to record proposal
- 12:30 - 13:30 - Lunch - local restaurants
- 13:30 - 15:00 - Session 7
    - Document proposal in preparation for a note - small groups working on markdown in github
- 15:00 - 15:30 - Coffee break– local cafes?
- 15:30 - 17:00 - Session 8
    - Summary of where we are up to, next steps

Things to come back to

- How do ObsCore (et al) queries work in a world where we have TAP 1 and TAP 2?

Session 5 - Talk preparation

Talk 1: Introduction to the protocol transition project

Slide 1:
Motivated by project calls at the last few interop meetings
IVOA standards don't cope well with modern rapid development tools

new projects joining the IVOA, new teams using modern tools
JF key essential missing element - technical (machine readable) description of the standard
Rigorous, unambiguous definitions
compatible with modern developer tools
"it just makes everyone's lives easier"

adopting the new format will be easier, it won't be starting over again
you don't have to adopt it now, but when you do it will be a lot easier

Change affects Everything that uses a Restful interface

Slide 2
How we got here

Slide 3
What we're doing - Open API; Split Std doc into 2 components

Slide 4:
Phases

- Phase 0 - Pilot project - Proof of Concept, deliver TAP, UWS WDs (Initial proof of concept for the tools and processes)
    - Text part of WDs minimal for start; prototypes that implement those things
        - fixing TAP API with current features and rewriting doc
            - New things in TAP would go into new TAP in phase 2
        - Decide which parts of OpenAPI we use and which we skip
        - Trimming the documents to remove the technical language that is covered by OpenAPI.
        - Informative of what the benefits will be
    - Initial proof of concept for the tools and processes
    - New style error handling
        - common style of error responses across all standards
        - draft note to cover all services
    - Derrived services
        - RegTAP, ObsTAP, EPN-TAP, ObsLocTAP, - need to check the implications, they all inherit TAP as-is and they don't change the REST API.
    - IVOA note based on experience
    - Note and WDs can be "very drafty"
- Phase 1 - first set of recomendations
        - TAP, UWS, DALI, service descriptors for DataLink
        - PRs and recomendations go thru TCG

        - figure out where the error handling goes (which document)
- Phase 2 other standards, when they need to be updated
    - New things planned for TAP - I would put these in phase 0
    - DataLink + parameterized services (parameterized services in DataLink using the new style)
    - VOSpace - keep monitoring to check that new changes won't break it

Slide 4
PROS; Process Improvements

Writing web service API in OpenAPI will mean DALI no longer needs to specific the details.
DALI changes

- form-style -> json-style
- interval - value types we would keep, but needs more serializations
- form-style string
- json-style data model for each field

SSAP - no plans to take it further, so it gets updated when we migrate it

Talk 2- Technical changes overview

It will be all right - honest
Simple to write and simple to read - using the YAML version
Using version 3.0 to be able to use the new tooling

Covers all of the REST services
what is changing
next version of a standard will have an open api definition
divided into 2 dcuments
narrative,
use cases
technical manual

- behavior

API spec

- If you need a line break in OpenAPI, then it should go in the technical manual
- As much as possible in the OpenAPI spec.

Inevitability
Moving away from fixed wire format towards data model based API.
Parameters defined as fixed types

Make it explicit we are not making a change to the XML schema of VOTable.

Identifying anti-patterns. If it doesn't ft OpenAPI, take that as a hint to make it better.
Example - case insensitive parameter names.

Immediate benefits

- modern tooling
- modern editirs
- better security

Smaller, simpler, text documents
e.g. TAP lists all of the endpoinst and parameters

automated testing
code covereage
CI/CD build checks for side effects

When we move to 3.1, we get modularization
no one is forcing you to adopt now.
not starting over
we will have tools to help
it will be the same standard, but easier to describe

new hires will find it easier to get started in the VO -
lower barrier to entry and less of a learning curve

eventually, the entire IVOA moving towards this

tom - one of the benefits is the compatibility with modern tooling
although the tooling is not required
but the benefits of automated code generation and testing
edit the API spec and test it interactivley

tooling is icing on the cake
seeing an interactive demo, showing people visually, brings it to life
Swagger Docs is a good example

Two ways to implement
Take the reference OpenAPI and generate a service automatically.
Write your own classes, generate the OpenAPI and compare that to the reference OpenAPI.

Coverage validator that uses the reference OpenAPI to check a service has all the required methods.

Improved consistency across implementations is an important message.

content negotiation

- XML mandatory ?
- JSON
- YAML

GM - we need to allow developers to use old tools to serialize original XML ?
DM - generating backwards compatible XML will be harder

Pat - CAOM repository service has VODML define content
can we trust the 'magic' to generate the right content
direct returns from the metadata API

Can we say the output will always be compatible

With VODML we have that already
Can this project specifiy how we serialise VODML
Can we serialse it as a big string ?

There are existing code bases that understand these data models
The payload interpretation is done using existing standards and tools.

New services can be 100% OpenAPI defined
Also legitimate to encapsulate existing data models as long strings
Same as we are planning to do with VOTable response.

VOTable response works.

PAT - at least one format needs to be mandatory

Example of serlization independent web service method
https://github.com/ivoa/CIRASA-planner/blob/1e2c43b4ea2c58a739a2934058f4508da024997b/experiments/pandak/src/main/java/uk/co/metagrid/pandak/PandakController.java#L212-L236

Backward compatibility was raised - can the auto generated repsonse be compatible with the old format? Probably best exp

Talk 3 - Implementation approach

consulting with people about the implications
in order to do phase 0 we don't require formal agreement from TCG etc

we will solicit feedback from the community

a new way to offer and formally define services and easier to deploy new ones
essential to the evolution of data services from data providers

and to continue to being ablt to support these protocols
with new technologies

not asking anyone to go back and re-implement existing services

in our community we have a large number of sevices scatterd over order 10+ data providers
smaller number of key clients (enumeration)
(do we need to mention the old astroquery tools ?)
(Gaia is still using the astroquery tools, and hasn't yet moved to astropy)
()
it is our preliminary assessment
that the key client implemntations have identifiable sources of funding
they are not orphaned and
are more likley to be able to take on this work

this is what it will mean for the community to adopt this process
it wil need the major clients to be prepared to do this work

Project adoption (eg. Rubin and SKA SRCNet) depends on having interoperable clients and services

yes it is work to be done, but it will be worth it in terms of
we are going to unlock creativity in the data publisher community
a better VO comes from having a lot of services adopt the new methods

once that work is done, we would expect the majority of the clients to still be able to use the old protocols and be able to use the new methods
we are not expecting them to change
there are 10,000's of services that are not likley to change, and we don't exxpect them to change
no duty for them to change

The ask is to have community involvment in development of phase 0.

Rubin, SKA SRCNet and CADC are enthusiastic about these new developments.
(CADC would implement new TAP by the end of phase 0)
(would Rubin contribute to Firefly development?)
(would SKA SRCNet contribute to client development?)
(SpaceTelescope are enthusiastic to proceed with phase 0)
"this house welcomes the efforts that have been out into developing a new framework for IVOA protocols .. "

there is a cost involved, but equally there is a cost to NOT doing this (outside discussion)

Firefly team will be busy with current work until after the next interop
PyVO might be a good case for collaborative development

the "coolness factor" of working with OpenAPI may mean developers may take it into their own time to experiment, but Firefly / Rubin can't yet guarantee developer time

if CADC TAP becomes available, and Rubin is able to use it, it may act as a motivator for raising the priority

the migration plan is to migrate the clients and use the new process to build new services
as resources permit for data publishers to migrate existing services
success does not depend on migrating existing services

CADC would have 2 separate services querying the same science database
the decision to decommision the older service is not a technical one

SpaceTelescope - envision a fairly long trasition period, monitoring the useage to see how long to keep the older services running

If phase 0 is successfly, Rubin would be deploying/developing new services with the new tech, and run both old and new versions of CADC TAP as long as they were required

if the MAST team were tasked with developing a new service, developers would be keen to use the new technical spec.
MAST team will be doing everythng they can to make these available

the definative part is phase 0 - and then we will decide where to go from there

Talk 4 - Implementor responses (more of a summary?)

- question on XML/JSON/other : choosing MUST serialisation vs. content-negotiation

Questions we know people arr going to ask
what the prototype is going to cover (or not)
about the process,

from the p3t wiki page :
"the aim of this group is to test the compatibility of the DAL protocols and by implication protocols such as UWS"
is this still valid ?
yes, we had to draw in DALI and VOSI because of their links to TAP

safe to say it is focused on DAL, but the intention is it will touch others in the future

the message:
phase 1 and then report at the next interop
Our ask to the comminity is lets explore phase 1 and see what the result looks like

open/seed questions
in general - questions and answers at the end to save us from getting diverted
note taker to collect questions during the session

designate note takers
alternate between us to cover the whole conversation
volunteers - Tom, Dave, Janet

Update schedule to show discussion at the end

"We'll have plenty of time for discussion at the end but are there specific unclear points to clarify now"

Value propositiion ot end user is more new services as they are easier to develop but does make some adhoc coding easier

Deliverable documentation

Unless otherwise specified, start as markdown on the GitHub repo.

Aim to create an IVOA note as final step.