Thursday 16 May 2024 - P3T day one

Attendees: In person - Joshua F, Dave M, Tom D, Marco M, James D, Pat D, Brian M, Gregory M, Jesus D, Janet E; Online: Russ A, Gregory DF

Draft agenda

- 10:00 - 10:20 - Setup
- 10:20 - 11:20 - Session 1
    - Welcome, ground rules, housekeeping, outline of schedule
    - Summary of what has been done so far - decisions made etc
    - Review key objectives
        - 1 - Agree on a detailed plan for changes to IVOA protocols so that they are compatible with modern web development tools (now and into the future).
        - 2 - Show how the Cone Search, UWS, and TAP protocols would be revised under this proposal
        - 3 - Produce content for the talks in the Wednesday P3T interop session
        - 4 - Commence work on content for an IVOA note describing the changes
        - 5 - Allocate people to talk to key implementors one on one about the changes
- 11:20 – 11:50 – short break
- 11:50 - 13:00 - Session 2
    - Demonstrations of the UWS, TAP, Cone search examples
    - Discussion on client impacts, provider benefits,
    - What still needs to be done?
- 13:00 - 14:00 - Lunch - local restaurants
- 14:00 - 15:00 - Session 3
    - How do we change over - discovery, transition
    - What tools are needed - validators etc - what can we use off the shelf and just guide people
- 15:00 - 15:30 - Coffee break – local cafes?
- 15:30 - 17:00 - Session 4
    - What does the VO landscape look like in 1, 5 and 10 years?
    - Test our transition plan against this

Summary of what has been decided so far

26 Feb

- Models - Russ
    - 'clean room' approach
    - Fast API coded and then generate OpenAPI model from that
    - Would need english description beside OpenAPI to capture all the nuance
    - Want to separate semantics from serialisation to make future tech changes easier
    - If we can't describe a thing in OpenAPI are we doing the right thing?
- Models - Joshua
    - More of a migration of current UWS to OpenAPI
    - Maintainability is important - so YAML would make sense
    - Have YAML doc separate to english spec and use CI to build the combined doc (or just side by side docs)
    - Could have static web pages (e.g.l swagger0 generated from yaml for browsing spec
- Try to seperate encoding from standard to make it model driven
- Aim to have example repo showing openapi alongside english spec

26 Mar

- Structured error handling
    - HATEOAS
    - HTTP error codes
    - Links to site with details of error
    - If desired provider can add custom entries and host those
    - Useful on sync, hard for async where request is being built up
    - DALI already advises 4xx codes
    - Want to get away from votables for errors
- Open API
    - Recommend use 3.0.2 as there is not sufficient generator support for 3.1.0 yet
    - Issue with modularity - quite a lot of repeated elements in a standard - would be nice to be able to use references
    - yaml expression preferred
- Desired Outcomes
    - UWS, TAP, cone search - define with openapi
    - Still discussing how close to stay to existing standard (level of change)
    - Expectation that old and new versions will need to coexist in VO ecosystem - likely some conesearch services will never change

26 Apr

- Structured Error messages
    - Allow both standard and implementation specific error responses
    - Could then store responses in a machine readable manner to later process and examine
- Structure for new protocols
    - Stick with HTTP as the base but don't exclude future change
    - Understand some services may go with e.g. Kafka+AVRO in transient-event space

    - Layers
        - Profile layer - HTTP+REST
        - Application spec - avoid HTTP-specific assumptions
    - OpenAPI
        - Might need different definitions for each payload format (e.g. XML or JSON)
        - However DM experiments have some classes producing both json and xml
        - Avoid forms based content to ensure modern security rules
    - Implementation
        - Expect clients to support new protocol alongside old and thus shield users from the change but need to engage with client authors
        - Each service provider can choose which version to support
        - Carrot would be that this will be easier to implement

Questions or discussion topics to come back to:

- Conflicts between narrative text and OpenAPI
    - discussed, Venn diagram shows limited overlap between the 2
- TD: Do we have consesnsus on whether we assume implementors will auto-generate code from the specs?
    - we do and user don't have to auto-genterate necessarily
- discovery transitioning (applies also to major REC changes)
- impact on DocStd and docrepo structure
- The term 'RESTful-JSON' has echoes of XML-SOAP from 2000's. So .. far the cost of doing both JSON and YAML with OpenAPI and Java Spring has been minimal.
    - Single doc works find to work rest api and data model of content - spring auto generates code with xml, json, yaml
- How will TCG RFC work with the "new" Proposed RECs? (enforce more implementation Notes?)
- ~~OpenAPI is a standard, how does IVOA recognises external standards?~~
- Should we specify the location where the openapi json can be found (like capabilities)
    - VOSI question

Thu morning discussion

Russ on 2 things he'd like to accomplish:

- Here is an arbitrary way of representing an IVOA protocol using REST and JSON
    - Removes many of the ambiguities that come from plain language

        - Lower the barrier of entry, both understanding and implementing the standard
    - Longer term: Here is a way of separating (wire) encoding from protocol specification

Are there orgs that do this kind of thing already?
K8s does this using OpenAPI specs

Note that this doesn't remove the narrative, just clarifies it.

DM WG does this a bit with the rigor and machine readablility of VODML with each model having an accompanying narrative.

benefits: validation

GDF: we will be facing 3 communities: service providers & developers, client developers, and users (richer ecosystem will show up for them)

GM: adding an OpenAPI doc to a specification, it's a new layer that we have to be careful it doesn't contradict the existing layers (DALI, ...) Avoid the "polygons" discrepancy seen in the past.

Conflicts between Narrative text and OpenAPI

JF: If there is a disagreement of the narrative with the OpenAPI spec, who wins?
Consensus: The OpenAPI version
Implications:

- We need to keep them in agreement. Though it surely there will be disagreements.
- If there is something you can't represent in OpenAPI, it could limit what you say (special exceptions, etc.)

Can we have flexibility
TD: In the past we have perhaps had too much flexibility and thus added ambiguity.
MM: Need to have both narrative and OpenAPI
TD: Words need to be there to describe the content where it doesm't belong in OpenAPI
PD: Need to avoid exceptions being added that are not compatible to the OpenAPI
JE: Keep in mind that the documents need to be reviewed by TCG - need to make it

[RA] Wire encoding - need 2 different - json+REST and either yaml or XML?
Think we're working towards four documents:

- A description of the separation between a service standard and a wire encoding, how to write a service standard and a wire encoding so that they work together, and anything that both parts need to be aware of, such as rich error messages, how to define errors in the standard and also allow local errors, etc.
- Ideally two separate wire encoding documents. One for REST JSON, and another one in some other encoding to test the design and make sure that the encoding is pluggable (YAML has been suggested, XML might be another possibility)
- A document describing an example service in the new style, including publishable OpenAPI specifications for both wire encodings.

Not sure what the natural next step towards that goal is in the IVOA process, but possibly a note laying out those goals?

[PD]

TOM - I disagree with that - the motivation is super important to let the machine readable to speak. Yes, it limits what we can say, but that is one of the problems that our current specifications have. In trying to be too flexible we have made them ambigous.

Marco The narrative the important

TOM - yes, the words have to be there to express the semantics, but we need to be precice.

JF - the OpenAPI spec covers 80-90% of what we can do in HTTP. If we end up requiring the 10% that isn't describable in OpenAPI, then perhaps we shouldn't.

Pat - yes, we have to be able to descibe it in OpenAPI.

What about the TCG review

Historically we transitioned from WSDL to REST without transition tools, so ended with hand-crafted REST specs. (Cautionary tale w.r.t. putting our eggs in the OpenAPI basket, though maybe transitions are easier now).
What about PDL (parameter description language)?

Parameter Description Language
https://ivoa.net/documents/PDL/20140523/index.html

Pat -we do need to keep the modularity of DALI and VOTable
JD - do we need our own tooling that includes cross referenced
TOM - there is an issue with one standard relying on a specific version of another spec.
Is there a distinction to be made for the content of a POST and GET that
JF - I don't entirley agree - on eof the reasons for not going for 3.1 was that the

tooling was not ready yet, but would be willing to sacrifice the extra tooling to be able to have the 'include' modularity of 3.1. Redices copy/paste errors.
TOM - I think we have to be careful about the benefits of strong typing, because there could be problems with versioning. Past experience with Java serialization. Trying to describe a VOTable response would be ... interesting
JF - concerned that we leave a door open for ambiguity
PAT - modularity for VOSI is probably fairly simple.
PAT - UWS, the TAP/DAL style, input parameters, will become more document based, like VOSpace.
PAT - Import UWS for the REST structure, and define the document content to describe the task in a separate data model.
G.DB have we disentangled the UWS and the job enough ?
does the payload have to be described in a new way

Discussion on reviewing and validating documents and the interreltionships between them
Ideal would be to have CI on the documents and then if an underlying change would break things then it would be flagged in tests

JE: IVOA has been focussed on documents and avoided code - maybe we need to change this and have more code involvement - needs to be an exec level change though

GDF: Agree. COnsequence of not having softeare focus is thaty software community does not take IVOA seriously. Avoid purely theoretical stdnmards, Been much better with mutliple implementations and validators recently. But that hasnt flowed to how the standards are seen. Been very fruitful for us to be working with pyvo maintainers - shows value of the standards . Also helps us to guide client implementation so they are made with understanding intent of the standards. WIll feed to CSP to assist wth more data in the ecosystem.

GDF: Important maturation for the community to confront software development. Is IVOA entering a new phase. Have a lot of standards and achieved a lot of the original vision.

[DM] Like the devops revolution of 'infrastructure as code', should we be moving towards 'standards as code' . Justification of the cost.

GDF we have many services running, and few clients (apart from invisible hand coded implementations)
We have a fixed list of clients TopCat Aladin, EsaSky Firefly, PyVO,
those clients will have to support old and new protocols
because there are old services that won't be updated
no requirement for active service providers like CADC to provide both old and new implementations
old inactive-development services can stay with what they have
new active-development starts clean on new protocol specification

TOM transition period, how we migrate. do we need a few more details about what we are proposing to do
by doing the first phases of this, just trying to describe the protocols using OpenAPI
are we just describing exisiting services as they are, and generate errata to fix the issues
or are we describing whole new versions of the protocols

JD We are here because the existing standards don't work in OpenAPI. Assumption is that they will need new versions of the standards.
examples case in-sensitivity of param names - fixed case will require a new major version of the spec.

Marco - mitigation of issues for older services, e.g. proxy libraries that convert old services into new services.

GDF DataLink is an example where it is hard to distinguish vision from the text of the specification. Text tends to decribe the details but no the vision.

Miro board from whiteboard: https://miro.com/app/board/uXjVKHUqPJs=/

Thu afternoon discussion

Registry will support showing if service is v1 or v2
UWS not in registry so would be picked up by adoption in specs that use it e.g. TAP, VOSPACE
Issue with major versions - registry doesn't store tap and v1 just tap-1

This work does introduce a dependency on the OpenAPI version - may need to evolve standards if the OpenAPI version changes.

To cater for this, include the openapi verison in the namespace e.g. tap-v2.0-openapi-3.0.yaml