Protocol Transitioning Tiger Team (P3T)

February Meeting - 26 Feb 2024 20:00 UTC

Attendees: James Dempsey, Sara Betocco, Dave Morris, Marco Molinaro, Russ Allbery, Tom Donaldson, Joshua Fraustro, Janet Evans, Jesus Salgado

JD: Russ - provided models; good discussion over the week from the group
Today's topic - discuss the models

RA: Russ's model - Overview - If came at it with little/no VO experience - what would I design to make it seamless with widest variety of web development tools??

RA: Chose to use fastapi code to generate openapi as it was faster for me than writing openapi directly - includes structured error messages; How to consider UWS attributes

RA: Don't want to pin protocol implementation to particular language/framework.

RA: Would need english language beside OpenAPI to capture all the nuance. YAML is trivially easy to convert to json, which is the format consumed by the tools. Important part is that we have definition that follows best practice (e.g. HTTP verb usage).

MM: using OpenAPI - most know and taken up. From previous conversation need to be careful about binding to a particular tool (e.g. openAPI). What about other solutions ??? Good to review - could see if one translate into the other.

JD: What are folks aware of from other standards for expressing APIs?
JF: not so aware of one. Real work is wrestling any standard into a specification.

MM: Action to check out other frameworks and what is out there - willing to look into it -- as a group! Write what you find to mail group.

JS: OpenAPI is currently the most widely used standard.

DM: openAPI & content negotiation. to be considered also when facing other possible frameworks (i.e. figure out what features do we want for our protocols)
DM: Should also exmaine how we encode things such as content negotiation

RA: Have seen these described but not sure if the code generators handle those so much. Want to seperate semantics from serialisation to make future technology changes easier.

TD: Trying to use OPENAPI approach to defining http interfaces; if we find things we want to do but OPEN API doesn't support - it's a checkpoint to ask 'why are we trying to do a particular negotiation that is not supported' Maybe we don't need to do it in a rigid way and reconsider. JD backed up TD comment - particularly need to keep an eye on standard HTTP verb use etc

JF prototype - a rough idea of what a standalone UWS would look like in OPEN API - defines end points, how to access, params, etc. Get it close given the effort and not combing through all of the details of UWS. Showed screen and described. Content at https://github.com/spacetelescope/vo-openapi/blob/main/openapi/uws/uws.yml

Error responses would be good to define

Document schema: have properties structure that can describe the fields of each document.

Dropped MAST OpenAPI URL into the swagger.io editor to show swagger docs

Benefit is that engineeer has the OpenAPI doc to refer to and doesn't need to interpret nuance of english descriptions

Using FastAPI to implement APIs and have a module to ease XML handling.

DM: YAML looks easy to maintain. Much easier than WSDL which was very hard to work with. We need to make sure these things are maintainable, so happy with this approach.

JF: Documents can be versioned, so can reflect multiple versions of the standard. Supports iterative versions

DM: If this were published by the standard could we generate test client to test any implementation.

JF: Yes, lots of tools. eg. swagger tools can generate client.

JD: Easy to read in YAML, but tools usually use JSON documents; is there a standard approach to translate from JSON to YAML?? JF: Yes, it exists e.g. swagger editor can output json but lots of tools.

TD: Versioning. A key requirement with minor revs of stds, need to be backward compatible (techniuqes to preserve). Are there any nicities that OPEN API can help in this area? With code generation , there is sometimes a step to rationalize the business side of it. What are other ppls thoughts with the versioning aspect??

JF: Can expand to expect something for 2 difference versions (of the standard) in the code. Each implementation decides how to deal with it.

JF: Benefit is the clear easy definition of the API

RA: YAML has a lot of complexities. May be some YAML constructors that we need to avoid. Can read YAML in standard libraries and outout json . May need to do some testing as we go.

TD: Some gotchas of having YAML in the PDF document

JD: With YAML best opiton is a YAML doc in the repo itself and then pull it into document itself

TD: Would be good to have CI on these to verify the YAML

JD: Yes we should set that up as part of this repo.

DM: Agreed.

JS: Do we need YAML in the PDF? Should we just include a pointer.

MM: Output will be way to handle these particualrly in the document submisison process. These need to be linked form the landing pages for each protocol. problem currently for e.g. VODML - can't find the XML from the protocol landing page.

RA: Caution on putting YAML in standard. Understand goal is have a complete standard but wouldn't recommend somone use the YAML from the PDF. Most useful if available for download as a separate file. More useful would be to have documentation sites (e.g. swagger pages). Would be good to have static web pages generated from the YAML (like google and AWS) but not sure about public tools. In Rubin have a Readthedocs entry for the readdoc entry. For the standard have to be careefula bout normative vs descriotive - have to emphasise that the YAML is normative and geerated docs are descriptive.

DM: Agree in response to RA; separate file and appendix; use to generate separate documentation.

DM: What are the things the implementors found were difficult in the industry standard tool?

JF: Very hard to figure out where the pain points are until you do the work, and hard to remmber where they were afterwards. Need to document the pain points as we go on the implementation.

RA: Documentaiton as you go is great. Wrote tech note. Great if other folks tried this too. Case insensitivity not expressible in OPEN API; Text plain responses without structured that are used for error messages; returning dates as text responses that are ISO formatted; Dates are interesting and important for ppl to know they are dates and not independent strings

JF: in UWS, "Some parameters may be updated via PUT request" - in standard but not specified which. Made it very hard to implement in OpenAPI. Much easier if it is specified in the openAPI doc.

DM: Minor point about dates- standards concentrate on astro dates. Need more flexible format for protocol interaction dates which may need time zone.

RA: Rubin standardises on UTC as have multiple time zones in project.

DM: Thinking we should be lenient on accepting time zones for dates coming in to protocols.

TM: Nervous about discussion on being lenient. Then puts onus on clients and providors to handle multiple things. Look carefully if we want to allow different formats.

DM: Take the point - suggest ISO format

RA: OpenAPI has way to tag string as a date. Then requires RFC 3339 IETF standard format.

DM: UWS long polling - how has this been for implementors? Assume it would be hard to describe.

JF: OpenAPI won't get into that - will just have wait parameter. However have had problems implementing wait behaviour.

RA: Kubernetes uses OpenAPI and autogenerated code and has similar wait behaviour. It also uses streaming so much more complex than UWS. Problem may be on the implementation - if not pub/sub concept then much harder to support.

JD: What do we do next?

RA: Converge on overall protocol - what type of changes would we want to make, problems we try to fix. Is where the biggest bang for buck is.

DM: start from common piece of YAML and write a client and server. Folks can go after different interfaces and see if they are interoperable.

JS: Do you use FAST API or use spring.
RA: FastAPI - mostly pure python
JD: Happy to help out on the spring implementation.
DM: Would be at least 2 weeks before could start
RA: interest in rust but would need to negotiate time.
JE: Yes, Rust would interesting as a way to see what is coming next

RA: Major competitor to OpenAPI is gRPC currently but very different way of

thinking. But don't think protocol direction has caught on. Worth thinking about how to write protocol such that could swap in gRPC instead of REST wihtout losing astro layer.

DM: One of the aims of the group to separate encoding from standard, so make it model driven, Good to aim for separation.

DM: How easy is it to include external file in OpenAPI/YAML. e.g.common error handling, or standards based on UWS. Do we need to repeat UWS section in e.g. TAP

JF: Easy to define response models in different file and import into main document. Swagger validator did complain. Works fine though. In repo have gone back to single file to make validator happy
RA: Open API v3 uses json references. YAML to JSON conversion may be a gotcha as not a YAML feature. YAML does not support references natively. Would also need to change name yaml to json when converting.

RA: Do we want to build concensus on UWS model, or error handling model.
DM: Would be interested in working on error handling. Would be good to have consistency in IVOA.
RA: What format to make documents in
MM: IVOATex is based on latex and have tooling to support it. Also IvoaTexDoc documentation.
RA: Happy with latex
DM: Start with email
RA: Can take an action to email out ideas on error handling based in part on rubin work.

JE: What will be the aim for Sydney?
JD: Example repository showing how the OpenAPI spec could be used with the english spec, including build tooling
JF: Continue with UWS - get it to a developed state
DM: Agree to have have it in shape of how it would work in a standards document, can be edited, show mechanics of how it would work for someone to edit alongside the standard

DM: What's the program for sydney
JE: Thu and Fri, see program. Have an informal disucssion for Sat on the program

RA: As Joshua has existing spec for XML, Russ has json, can check transition between the two. Could be another good output. See what transition looks in practice.

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

- 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

Next meeting

Mon 25 Mar 20:00 UTC