JSON Splinter Discussion

29 Apr 2022, 14:30 UTC

gather.town: https://app.gather.town/app/PuCDQczUl8pfyXH3/ObAS
15-20 attendees

From the topic description in the hack-a-thon pad:
* JSON tooling is improving all the time whereas XML tooling is fairly static.
* Should IVOA start planning a future where JSON is supported? Is the intent for IVOA to still be requiring XML in all services in 20 years?
* Can interested parties discuss (1) whether IVOA should do this (2) what the main blockers are for starting to experiment?
* This also brings up the issue of whether we want to switch to a more REST-based services API. We are still using the "best practices" approach of 2004.

It would be good to be clear which parts we're discussing:
    * Service metadata (e.g. /capabilities end points)
    * Data metadata (e.g. metadata about a table)
    * Actual science data (e.g. the output from a TAP query)—the absense of a distiction between int/float, and the absense of Inf/NaN in JSON needs some discussion—https://seriot.ch/projects/parsing_json.html and the associated GitHub repo is a good resource around other potential pitfalls

Frossie making a case that the data exchange format (esp. JSON) is something we should be addressing as a community. Has a few introductory slides:
https://wiki.ivoa.net/internal/IVOA/InterOpApr2022Apps/vo-tng.pdf

Current web frameworks like FastAPI implement web APIs with strong RESTful semantics
- security is a factor
- bearer tokens
Making a case that transition may not be as hard as we default to tdhinking.
Mentioned scalability. [not clear to me how JSON addresses that]d
Obviously this a breaking change. So maybe we embrace change. "VO 2.0"

MD: Out of curiosity: Has anyone had real security issues? Of course, XML *does* have the problem that you can entity-bomb it, so this is something I find borderline convincing -- Markus.
We were vulnerable to entity-bomb (a spectacular simple attack), but it never actually happened.

MD: +1 on getting rid of case insensitivity whereever we can.

Well, we cannot change the schema namespaces (to https) without breaking everything. But then, clients shouldn't retrieve the schemas in normal operation. Of course, that's moot if we drop XML...

MT: One advantage for staying with XML is stability.
FE: Benefit of more current and web native outweighs benefits of stability. Would help with collaboration among those with funding for these efforts that are otherwise using more recent technologies/customs.
MD: Of course, in five years, the fashion in the mainstream will again be something else... Do we move on then, too? From a registry perspective, I'd be very skeptical we could pull that off.
TD: Yes for sure! Does this mean we should stay stable, or be better at adapting to change?
MD: The problem is: it's not *us* that need to adapt, it's our users and deployers, who may not appreciate us making them more adaptable :-)
TD: Yes, I don't know how to find that balance. It is the (costly) reality in much of tech and web services though.

Although being a big JSON fan (and user), I've to admit that I failed to convert VOTable XSD into a JSON schema.

Anne R. [AR]: I'll also note that a large part of the Planetary Data System (PDS) decision to go with an XML implementation was based on a need for very rigorous validation of metadata. 15 years ago, at design time, JSON could not provide that tech. So the primary implementation was XML and JSON translation of the metadata structures was provided as a service. The translation from our full XML-implemented Information Model to the JSON structures was necessarily lossy - there were not enough structures in JSON to represent the subtleties supplied by the XSD standard. JSON Schema was not yet a thing. It is now, so it is entirely possible that over the next 10 years, we will migrate the primary implementation to be based on JSON schema and related validation. We did, at least, set up our Model and system so that we can create a new implementation while the old implementation continues unaffected, and we can change things under the hood. As long as we have equivalence, we can add a translation layer for legacy applications while moving forward with new implementations. Or at least, that's the theory...

JT (from chat): We (Data Central) use DRF (which is somewhat similar to fastapi), and find that autogenerated JSON is a pain, the NaN/Inf issue has bitten us so many times
TD: I agree about auto-generated code, and find the NaN/Inf issue to be a frustrating limitation to JSON.

Interesting follow-up discussions on Slack starting from these messages:
    #mtg-virtual-2022:
https://ivoa.slack.com/archives/C03CLG96C4S/p1651246127884569

#applications:
https://ivoa.slack.com/archives/CDYNRNBNF/p1651245116957869