JSON Information System, SRR/CRR

How did it get here, where is it going and what's it all about?

Ste Jones, Liverpool

GridPP43

August 2019

Topics

- I'll talk about what I've learned and try to answer some of these questions.
  - What is it?
  - Who is doing it?
  - Why might we need it?
  - What will it do?
  - How will it work?
  - How do we know it's right?
  - What else needs to happen?
  - Etc.

Introduction

- At the WLCG level, there is an on-going effort to produce an alternative to the BDII.

  https://twiki.cern.ch/twiki/bin/view/EGEE/WLCGISEvolution

- This system will run in parallel with the existing BDII at first.
- It will a simpler data design based on use cases (and of course it uses modern technology.)

  https://twiki.cern.ch/twiki/bin/view/EGEE/WLCGISEvolution#Mandate_and_Goals

- At present, the work has focused on static elements of the information system.

Introduction

- Other participants in this effort are here – Andrew McNab and Alessandra Forti.
- They've had a lot to do with the modelling.
- AF's talk on the workings:

  https://indico.cern.ch/event/721185/contributions/2964544/attachments/1636659/2611523/2

Why?

- Who thinks the BDII is:
  - Accurate? Reliable? Up to date?
  - Easy to configure? Easy to use?
  - Maintainable? Coherent? Relevant?
  - Adaptable? Efficient? Consistent? Easy to read?
  - Easy to query? Self-explanatory?
  - Seamless? Well-written? Documented? Testable?
- Well, that's why, then...

Why?

- Who thinks the BDII is:
  - Accurate? Reliable? Up to date?
  - Easy to configure? Easy to use?
  - Maintainable? Coherent? Relevant?
  - Adaptable? Efficient? Consistent? Easy to read?
  - Easy to query? Self-explanatory?
  - Seamless? Well-written? Documented? Testable?
- Well, that's why, then...

Technology

- New Information System uses JSON to model static site elements.
- At present, after much iteration, viable models have been agreed for
  - COMPUTE RESOURCE RECORD (CRR) - describes a site's compute.
  - STORAGE RESOURCE RECORD (SRR) - describes a site's storage.
- The models will be represented in JSON and served by a site's web server over http.

Technology

- The urls for a site's SRR and CRR will be listed in GOCDB under the site's entry, as Extension Properties - a name/value store for extending GOCDB in arbitrary ways.
- Eventually, dynamic data will be included. The plan is to have a particular element (SE, CE, whatever) serve its own dynamic data model.
- It's literally Object Oriented.

Progress

- I was more involved some years ago; I wanted the CRR (Computing) to use, as its primary entity, the Batch System, not (as with the BDII) the CE.
- Then I did something completely different for at least a year (HTCondor-Ce testing/ documentation + APEL accounting) and then I came back to it. There had been a lot of thrashing around with the schemas, entities, names, multiplicities, fields…. modelling stuff.

Progress

- So when I came back to it, I was thrilled that the model that the WLCG Information System Task force had settled on did make the Batch system the central entity! I had accidentally won an argument. It does not get better than that.
- Since then I've been mostly involved with Automatic Validation of the JSON files. I'm trying to stay out of politics, since the models are more or less structurally OK I think and I don't want to meddle, having not participated much so far.

Need for automatic validation

- When I was somewhere else, draft manual schemas for SRR and CRR had been written up. Several sites in Europe/UK had issued various example JSONs which reputedly conformed to the manual schemas.
- But, to say for sure if the example JSONs were valid, we have to take out the ambiguity. One off comma makes a JSON not well formed. One additional or misspelled tagname makes a well formed JSON unparsable. These aspects had to be automated to make the problem "tractable".

Automatic Validation Implementation

- Zealots say that JSON is not like XML. But JSON is like XML, since it must be well-formed. If it's not well-formed, it's (basically) not JSON.
- JSON is like XML, since it must conform to the relevant schema. If it does not conform to the schema, it cannot be parsed normally (by a machine.)
- JSON is like XML since even if it is well formed, and schema compliant, it still may not make sense!

Automatic Validation Implementation

- In particular, schema validation software does not at present check for name uniqueness or referential integrity issues.
- To know that it's right, a test parser must be written to enforce these extra integrity rules.
- All of this is done with open source Java or Python libraries.

Automatic Validation Implementation

- Prototype technology to perform automatic JSON validation has been developed and is under version control at:

  https://github.com/sjones-hep-ph-liv-ac-uk/json_info_system

- It has these parts:
  - JSONSchema schemas for CRR (compute) v1.5 and SRR (storage) v4.2. These use version 7 of JSONSchema.
  - Java to validate a JSON (whether well formed and in compliance with the relevant schema.)
  -

Automatic Validation Implementation

- More parts:
  - Java to parse a valid JSON to do further checks related to data integrity (unique names, valid relationships...)
  - A website that allows a user to post a JSON file for validation, returning a status and description.
  - A RESTful webservice that does the same validation in a way that can be scripted with (say) curl.
  - Some equivalent work in Python that might be used on the command line (incomplete.)

Automatic Validation Implementation

- The website is here:

  http://hep.ph.liv.ac.uk/JisValidator/JVMain.jsp

- The current options are to test a CRR or an SRR JSON, or to view the schemas.
- You can select whether to do an optional integrity check.
- You can select which version of the appropriate schema to use for the validation phase.
- Code needs hardening, esp. for integrity check.

Automatic Validation Implementation

- The webservice can also be used; here are some examples to check storage and compute JSONs.

curl -i -F jsonfile=@/root/storage_service_v4.json https://hep.ph.liv.ac.uk/JisValidator/rest/jsoncheckws/srr

curl -i -F jsonfile=@/root/liv.json https://hep.ph.liv.ac.uk/JisValidator/rest/jsoncheckws/crr

Automatic Validation Implementation

- You can specify the schema version and whether to check JSON integrity in both Browser and CLI/RESTful interfaces. The RESTful interfaces take the ver and integrity parameters, e.g.

```
curl -i  -F
jsonfile=@/root/storagesummary_lanc_edtowork.json
https://hep.ph.liv.ac.uk/JisValidator/rest/jsoncheckws/
srr?ver=4.1\&integrity=yes
```

Formal Schema Defintion

- The SRR/CRR schemas create a formal structural definition of a JSON (together with the integrity checker, to imposes extra rules).
- If an SRR/CRR does not make it past the validation, it's GIGO. We can't have it.
- I want to persuade the task group to use the formal schemas as the “Primary Definition”, and to regard the manual schemas as an aid to comprehension. We can put the formal schemas in version control and treat them as a software product.

Further work

- Use the tools to wring out faults and ambiguities in the formal schemas themselves, the JSONs and the manual schema definitions
- An on-going maintenance task to keep schemas up to date and adapt to new requirements.
- Dynamic JSON provider elements need to be developed.
- Obviously all this needs to be integrated into downstream consumers, CRIC, REBUS have been mentioned…

Live demo!

- Check this SRR JSON

  https://hep.ph.liv.ac.uk/~sjones/GridPP43/prague_dpm.json

- With the validator

  http://hep.ph.liv.ac.uk/JisValidator/JVMain.jsp

Questions? Ideas? Suggestions?

Criticisms? Insults?

….