4 - GA4GH Connect 2022

GA4GH Cloud API Implementers Workshop

Date: Wed, Apr 20, 2022
Time: 2:45p - 4:15p EDT
Slides: Link
Meeting Chair(s): Lee Pang (Virtual)

	Agenda Item	**Speaker**	**Time**
1.0	Introductions and session overview	Session Chair	2:45p
2.0	Implementation talks	Invited speakers	2:55-3:45p
2.1	GA4GH WES and the Amazon Genomics CLI	Lee Pang	3:00
2.2	GA4GH WES at DNAStack	Patrick Magee	3:10
2.3	GA4GH WES for Cromwell	Sara Salahi	3:20
2.4	GA4GH TRS and WES in Dockstore	Denis Yuen	3:30
2.5	buffer	n/a	
3.0	Discussion, issue generation, Session summary	Group	3:45-4:15p
4.0			

Session Participants

Speakers:

- Lee Pang (Virtual)
- Brian O'Connor (Virtual)
- David Glazer (In-person)
- Patrick Magee (XX)
- Sara Salahi (XX)

Session 4: GA4GH Cloud API Implementers Workshop

Wednesday, April 20: 2:45 - 4:15 pm EDT

Printemps

Zoom Link:

https://us02web.zoom.us/j/6971797978?pwd=dXlCazNWOG5Db3VUcEVybG9yTHdFUT09

Collab Tool Link:

Best Practices for Virtual Speakers

- Make sure laptops are plugged into power prior to and during your session.
- Check cameras to ensure they are centered, sit in a well-lit area and ensure your background is what you want for your presentation.
- Use an ethernet cord for the best connectivity; if using Wi-Fi, make sure to test your Wi-Fi connection prior to the conference to ensure it works.
- Earbuds or headphones will prevent audio echoes.
- Please stay muted except when speaking.
- All sessions will be recorded, and the chat boxes will be saved.
- Have water or a beverage close by.
- For any technical issues the day-of the meeting, please contact Ida Donner, idonner@palladianpartners.com.

Onsite Support: Neerjah Skantharajah

- Meet with A/V technicians 10 mins before to load slides and collaboration tools, go through session agenda (run-of-show) so they know when slides need to be presented, when to switch screens, when it is a virtual presenter vs onsite presenter, etc.
- Ensure A/V technology is working (mics, projectors, etc)
- Help pass mics if required
- Join Zoom with personal laptop to moderate sli.do

Onsite Zoom Moderator: Angela Page

- Start the Zoom meeting 10 minutes before
- Start cloud recording
- Chat the link to agendas and other meeting materials throughout the meeting
- Monitor chat for questions; read them out loud or help flag them for chairs to read out loud.
- Help participants mute and unmute
- Remind participants to raise hand to speak

Session Notes

Attendees:

Lee Pang (AWS) (chair, virtual) David Glazer (Verily), Patrick Magee (DNAstack),,,,,,,Denis Yuen (OICR - Dockstore),Shaikh Rashid (CanDIG - UHN),,,,,,,,,

Github Repos:
https://github.com/ga4gh/tool-registry-service-schemas
https://github.com/ga4gh/workflow-execution-service-schemas

Minutes

Amazon Genomics CLI [WES] (Lee Pang)

- Uses GA4GH WES to communicate with multiple different workflow engines
- primary goal: aid in adoption of cloud for genomics, including migration from on-prem
- heard from researchers using WDL, Nextflow, CWL, SnakeMake
- used OpenAPI-generated clients; worked for Python; needed to hand-edit the Go version
- needed to build several WES->engine adapters; about the same effort as it would have been to write a CLI->engine adapter directly (but now reusable)
- see slides for detailed “challenges and pain points” and “suggestions for improvement”
    - PRs are in flight for many of these

DNAStack [WES] (Patrick Magee)

- see slides for detailed Challenges and Suggestions
- challenge: had to invent ways to pass auth between WES and DRS/TRS/…
- challenge: not enough info to build a UI or other tooling
- suggestion: workflow engines are different – have to understand engine-specific idiosyncrasies

Cromwell [WES] (Sara Salahi)

- investigating adding WES support to a Cromwell App; not implemented yet
- concept is offering a single submission service using WES to call {Cromwell, Nextflow, …} engines
- next milestone is submitting wf’s from submissions service via WES to Cromwell App on Azure – aiming for two quarters out

Dockstore [TRS, WES] (Denis Yuen)

- OpenAPI-generated Java client needed some hand-editing
- differences between WES implementations - AGC vs. DNAstack have different Run Request payloads
- NOTE: the “endpoints are slightly different” issue is not quite the same as “engines are slightly different”, but closely related – the leakiness leaks up

Discussion

- Patrick:
  - Hone in on a specific auth mechanism in the spec
- Other apis?
  - DRS is an easier problem to solve than WES which has to deal with more opinionated engines
- How can we use these apis for production?
  - Patrick: DNAstack uses them internally
  - Lee: WES is at least really close – we’ve injected some of our opinions into how to use the spec, which should be codified as part of the spec, but given that it’s pretty solid
- Suggestions for WES
  - Address engine leakiness
    - Augusto: Need to know details about the execution environment
  - Almost sufficient to build an application that only communicates via WES
- Suggestions for TRS
  -
- Suggestions in general
  - How to enable building UIs?
  - How to manage requesting versions of things, timestamps
    - Action to create issues for TRS and WES
- Q: important engines/languages?
  - Lee: WDL, very closely followed by NextFlow, then SnakeMake, then CWL
  - Lee: Cromwell, NextFlow, SnakeMake, then also miniwdl, exploring CWL
  - Sara: the same list for us; we haven’t prioritized, but probably NextFlow after Cromwell
- Conversation to take to roadmapping:
  - What are the next release items for WES in particular?

Session summary

Implementations by AWS, DNAstack, Broad/Cromwell, and Dockstore were presented showcasing expectations, successes, and challenges for WES and TRS APIs. All speakers commented on WES with Dockstore also speaking about TRS. Most of the discussion focused on how to improve WES. Guidance for implementing auth and interoperability with other GA4GH APIs was called out for both WES and TRS. Challenges and suggestions for WES fell into two primary themes - increasing the prescriptiveness of the WES API and improving the developer quality of life for building applications that utilize the WES API by filling in gaps in capabilities and request/response details. Several specific WES issues were identified and will be added to the WES repo. For TRS there were no significant challenges faced by implementers and only enhancement suggestions were called out.

Successes:

- WES is a small API that is relatively easy to implement and add custom functionality to (e.g. custom auth). Proven capable of functioning as a common API for multiple workflow languages (albeit with some rough edges - see challenges)
- TRS has no callout challenges

Challenges WES:

- Existing WES implementations differ in subtle ways due to differences in spec interpretation and expectations for auth, request parameters like workflow_params and workflow_engine_parameters, how to handle workflow descriptors and attachments. This adds difficulty in creating tooling and UIs intended to work with WES.
    - Auth open to interpretation - implementations differ from static tokens to dynamic (e.g. AWS SigV4)
    - Not enough clarity or capability in spec (e.g. list runs) to build a UI with good UX
    - Wider compatibility with multiple workflow languages - or be workflow language agnostic
- Developer QoL:
    - OpenAPI generation doesn't always produce functioning code (Go, Java reported as having issues). Spec doesn't enable generating code for auth.
    - Difficult to support multi-tenant deployments.
    - Run listing requires state preservation during pagination.
    - Requirement for JSON serializable request parameters and responses

    - Cannot rerun workflows based on RunLog alone. Not enough information captured - e.g. attachments associated with specific run-id
    - Limited or no mechanism for endpoint capability discovery
- Poor interop with other GA4GH APIs (e.g. DRS, TRS and how to pass auth information)

Suggestions WES:

- Prescriptiveness:
    - Work backwards from what a workflow runner needs rather than a common slice of what engines currently support
    - Figure out what the user needs and make sure the spec checks most of those boxes.
    - Guidance and reference for how to do Auth is needed
    - Simplifying the URL structure and enabling clients to retry sending workflows without worrying about running them twice. Recommendation: Make clients specify IDs in requests
    - Allow only one way to submit a request*, e.g. either URL to descriptor or descriptors inlined (or clarify that implementations support both) with format for input standardized. Possibly remove workflow_attachment altogether.

- Scalability and Developer QoL:
    - Focus on improving conformance and uniformity across WES implementations
    - Eliminate vagueness. Add more clarity to request parameters and response structures
    - More details in API responses - allow WES API alone to be used for applications
    - Focus on automation and discoverability of configuration. There is currently no way to form a WES request without a priori knowledge of the workflow and engine itself making it a challenge to build tooling.
    - Enable lower latency in responses - e.g. add a paginated endpoint for workflow tasks (e.g. /runs/{id}/tasks). Suggest adding {id, state, resources} to items
    - Allow for tasks and workflows to fail unusually - e.g. empty exit_codes
    - Don’t let engine implementation specifics leak into the API
    - API (e.g. /service-info) should clearly indicate what is supported

Suggestions TRS:

- New tool classes (API definition is open-ended) can differ
- New workflow languages (enum, issue opened to describe available languages via service-info)
- Adding write features
- Optional authenticated behaviour

—

Wes issue:
Originally, implemented for a multi-tenant system and we tried to stay true to the then requirement of having /ga4gh/wes/v1 at the root of the path. This did not scale
Recommendation: Allow WES api's to be defined at an arbitrary level of the path

Wes issue:
Listing runs technically requires state to be preserved between pages. This was technically impossible
Recommendation: do not require this in the spec

Wes issue:
Not enough information available when listing runs to build a UI or other tooling.
Recommendation: Add additional properties to the RunStatus object

Wes issue:
No filtering supported on Runs, meaning a user would need to list all runs to drill down to what they were interested in

Wes issue:
RunLog implies that the workflow_params and outputs are JSON serializable

WES Issue:
Recommendation: Switching /cancel from POST to DELETE would allow us to use the same request URL with different verbs for all of POST, GET and DELETE.

WES Issue
Remove the workflow_attachment bundle since for engines like Cromwell forwarding to GCP or Azure it makes no sense to accept input files via the API.
Recommendation: Assume every script, reference, and input should be fetched at runtime, not uploaded at request time.

WES Issue
Add workflow labels and ability to identify priorities

WES Epic
Very poor interoperability with other GA4GH apis. We needed to invent ways to pass information on how to authenticate with DRS, TRS and other GA4GH services.