Interfaces Update

Mike Conway
Data Systems Architect/Engineer
Deep Patel
Software Developer
National Institute of Environmental Health Sciences
Developing a Commons to manage research data, using iRODS as a platform for unifying and managing local and cloud resources.
Data Commons integrated with processing pipelines and workflow systems.

Use Case:
- Data Commons as the hub for managing research projects in an ISA model
- Sample submission integrated with Clarity LIMS triggers NextFlow pipelines
- Data Commons as delivery mechanism gathering metadata and pipeline results

Setting future strategy anticipating move to cloud over time, with a hybrid of local research data, published artifacts and tiered storage in the cloud.

How can we develop strategies that work for cloud and local use cases?
GA4GH Cloud Work Stream APIs

Sharing Tools and Workflows

Executing Workflows

Executing Individual Tasks

Accessing Data

https://docs.google.com/presentation/d/1_MFTCw1uDrFntbki2Nvyh2I2lYOlQKTHmrZgMTspdm4/edit#slide=id.g54dc8a46d6_0_0.
What interfaces?

- Web Interfaces
- REST API
- Standard protocols
Web interfaces have been a tough road

- Hard to get a consensus
- Hard to find resources
- Hard to develop something that is generally accepted/useful
Metalnx

- iRODS Web Interface
  - Released into open source by Dell/EMC
  - [https://github.com/irods-contrib/metalnx-web](https://github.com/irods-contrib/metalnx-web)
# Plusses and Minuses

<table>
<thead>
<tr>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is a thing that finally has some small amount of consensus</td>
<td>It’s not X,Y,Z</td>
</tr>
<tr>
<td>Out of the box it has lots of interesting functionality, especially in administrative tools and user functionality</td>
<td>It has an opinion on various administrative and user functions that are not well-defined in core iRODS</td>
</tr>
<tr>
<td>Has many standard constructs and frameworks common in the enterprise</td>
<td>Mixed approach (page rendering versus AJAX) that cause occasional inconsistencies</td>
</tr>
<tr>
<td>Has actually been super-stable</td>
<td>A bit hard to initially deploy and configure</td>
</tr>
<tr>
<td>Provides a good starting point</td>
<td>Has some amount of technical debt to manage</td>
</tr>
</tbody>
</table>
Overall Strategy

● Initial simplification and draw-down of scope
  ○ Hide some functionality for now (e.g. Dashboard)
  ○ Migrate functions specifically implemented as a Metalnx service with database support into iRODS core server facilities
    ■ E.g. looking at group and user management
  ○ Simplify app structure
    ■ Extract ‘services’ and recast/refactor as standard extensions interfaces
      ● Make services more universal
      ● Allow gradual refactoring underneath
      ● Minimize circular dependencies
Overall Strategy

- Migrate existing features to more widely accepted/applicable features
  - Extract features one-by-one and re-assess through community processes
    - E.g. **Metadata Templates** from an internal Metalnx and database implementation to a formal working group and a Python rule engine focused service
  - Consider essential non-core functionality and refactor
    - Move towards OpenAPI and Microservices
    - Make features configurable and language independent (also alleviates classloader issues in core)
Overall Strategy

- Try to use Metalnx as a way to bring in community innovations and shape as widely applicable features
  - Encourage migration to a microservices architecture
    - OpenAPI and focus on multi-client
  - Make additions optional and configurable
    - E.g. Notifications
Overall Strategy

- Use Metalnx as a tangible basis for discussing new client capabilities and patterns
  - Identify common user expectations and how they can be mapped to new capabilities
    - e.g. new Pluggable Search

- Migrate defined abstract interfaces into REST endpoints
  - Go piece by piece, over time, surfaced in Metalnx
Developer Workflow Enhancements

- Collapse Metalnx codebase into a single project
- Extract common services out, begin folding into abstract interfaces, remove circular dependency problems
- Move towards a .war overlay for site-specific skinning and enhancements
- Migrate services into pluggable, adaptable microservices with a wider client base
  - JWTs
  - OpenAPI Specs
  - Containers
- Maximize capabilities as iRODS.THINGS and minimize Metalnx.THINGS, especially irods-ext database
- Move to Vue.js over time
Notifications

- Based on CyVerse API (https://github.com/cyverse-de/cyverse-de.github.io/blob/master/api/endpoints/notifications.md)
- Two initial use cases:
  - Notification of data delivery via processing pipelines to PIs and data requesters
  - Notification of data access requests and approval/disapproval
Notifications

Data Commons Notification API
This is a notification microservice for NIEHS data commons.

Contact the developer
Apache 2.0

<table>
<thead>
<tr>
<th>Method</th>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>/notification/delete</td>
<td>Delete notifications</td>
</tr>
<tr>
<td>POST</td>
<td>/notification/messages</td>
<td>Add new notification</td>
</tr>
<tr>
<td>GET</td>
<td>/notification/messages/{userId}</td>
<td>Get all notifications</td>
</tr>
<tr>
<td>PUT</td>
<td>/notification/seen</td>
<td>Update notifications as seen</td>
</tr>
<tr>
<td>GET</td>
<td>/notification/unseen-count/{userId}</td>
<td>Get count of unseen notifications</td>
</tr>
<tr>
<td>GET</td>
<td>/notification/unseen-messages/{userId}</td>
<td>Get all unseen notifications</td>
</tr>
</tbody>
</table>

[BASE URL: /v2, API VERSION: 1.0.0]
Example from DE
Metadata Templates

- Define templates as JSON Schema
- Bind templates (required or suggested) to collections
- Present curation tools for templates
- Store metadata as AVU
- Validate and manage exceptions
- Retrieve as merged templates for editing and display
- Base new search tools on templates
Search

- Directly leverages the now-developing indexing capabilities
- Allow custom search schema (initial target ElasticSearch schema) to be plugged into Metalnx and other iRODS clients, even iRODS itself
- Inspired by PubMed Interfaces
PubMed Model

- Select a provisioned search schema
PubMed Model

- Schema defines applicable attributes for text search or advanced search builder
PubMed Model

- Search can have a familiar ‘Googleish’ result format that can be customized by schema
Search API

See https://app.swaggerhub.com/apis/michael-conway/data-grid-search-api/1.0.0-oas3
Search API implementation responsibilities...

- Describe the schema supported at this endpoint
- Describe the attributes supported by this schema
- Accept searches in the following formats:
  - Free text “glioblastoma cell gene”
  - Free text with attributes “CELLLINE:xb4-124 ASSAY:RNA-Seq”
  - Builder style attributes
Search API implementation responsibilities…

- Interpret various searches and translate into appropriate query (E.g. ElasticSearch DSL)
  - Search hypotheses and project description as free text with fuzzy matching, search SampleID as an exact match
- Return queries in one of two formats
  - Google like result
  - Virtual Collection (ILS Listing)
Configuration in Metalnx

# Pluggable search configuration. Turn on and off pluggable search globally, and configure search endpoints.
# N.B. pluggable search also requires provisioning of the jwt.* information above

# configured endpoints, comma delimited in form https://host.com/v1
pluggablesearch.endpoints=
# enable pluggable search globally and show the search GUI components
pluggablesearch.enabled=true
# JWT subject claim used to access search endpoint for data gathering. User searches will utilize the name of the individual
pluggablesearch.endpointAccessSubject=
# timeout for info/attribute gathering, set to 0 for no timeout
pluggablesearch.info.timeout=0
# timeout for actual search, set to 0 for no timeout
pluggablesearch.search.timeout=0
Metalnx Responsibilities

- Based on configuration, initialize and ping each endpoint to gather registration information (schema/attributes)
- Cache schema info to provision search interfaces
- Call search and present results
  - Google like result will include links to objects/collections
  - Virtual collection row/column format can be used as a familiar collection display
Example Implementation

- https://github.com/michael-conway/grid-search-service is an example
- There are many opportunities to add frameworks and abstract class support for development!
- Jargon-extensions-if contains support for adding search to other types of clients, REST API:
Streaming

- REST
- Metalnx
- Cyberduck
- NFS
- WebDav
- CIFS
- GA4GH DRS

All rely on streaming i/o
The most important client-supporting move the Consortium needs to make is a dedicated effort to **extract maximum performance out of streaming i/o**!

Maybe even more important that pluggable multipart transport, or could even be a key enabler!
Thank You!

Mike Conway
Deep Patel
NIH/NIEHS
Office of Data Science

https://www.niehs.nih.gov/research/atniehs/dntp/osim/index.cfm

mike.conway@nih.gov
deep.patel@nih.gov