# Capabilities
# Indexing and Publishing

Jason M. Coposky
@jason_coposky
Executive Director, iRODS Consortium

June 25-28, 2019
iRODS User Group Meeting 2019
Utrecht, Netherlands

# iRODS Capabilities

- Packaged and supported solutions
- Require configuration not code
- Derived from the majority of use cases observed in the user community


Storage Tiering


Auditing


Provenance


Data Integrity


Automated Ingest


Indexing


Compliance


Publishing

For example - Storage Tiering

- Data Access Time
- Identifying Violating Objects
- Data Replication
- Data Verification
- Data Retention

The storage tiering capability - implemented as a composite which delegates each requirement out to separate policies.

# Policy Composition and Capabilities

Policies composed into a Capability framework delegate by naming convention:

- irods_policy_access_time
- irods_policy_data_movement
- irods_policy_data_replication
- irods_policy_data_verification

Each policy may be overridden by another rule engine, or rule base to customize to future use cases or technologies

Each policy may now be reused and combined into new Capabilities

# Indexing

A policy framework that provides an asynchronous, scalable full text and metadata indexing service driven by collection metadata

- Indexing technology of choice is reached by delegating policy implementation
- Document Type identification is delegated to a policy invocation

# Indexing Policy Components

- Document Type
- Indexing Policy Implementation
  - irods_policy_indexing_object_index_<technology>
  - irods_policy_indexing_object_purge_<technology>
  - irods_policy_indexing_metadata_index_<technology>
  - irods_policy_indexing_metadata_purge_<technology>

<technology> is directly derived from metadata and is used to delegate the policy invocation

# Indexing Overview

iRODS

The iRODS Indexing Capability provides a policy framework around both full text and metadata indexing for the purposes of enhanced data discovery.

Logical collections are annotated with metadata which indicates that any data objects or nested collections of data objects should be indexed given a particular indexing technology, index type, and index name.

From the configured metadata, the framework composes a rule name and then delegates to the policy implementation through the rule engine.

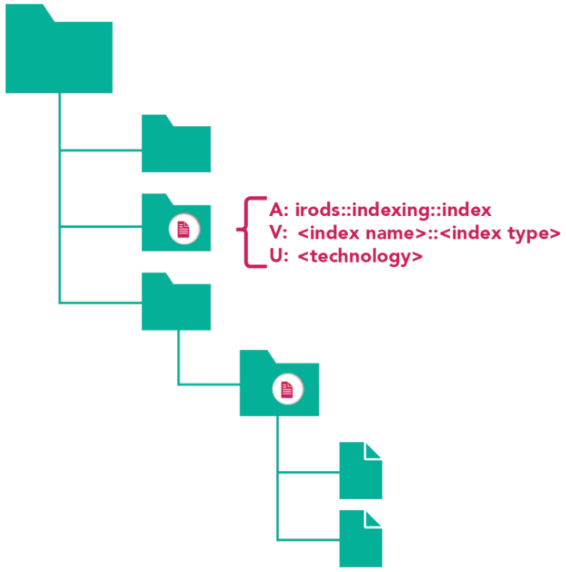
A new indexing technology can be supported via a rule base or policy engine which provides policy implementations of the form:

- irods_policy_indexing_object_index_<technology>
- irods_policy_indexing_object_purge_<technology>
- irods_policy_indexing_metadata_index_<technology>
- irods_policy_indexing_metadata_purge_<technology>

## UNIFIED NAMESPACE

Metadata takes the form:

- <index name> is the name of the index created
- <index type> is either "full_text" or "metadata"
- <technology> is the targeted indexing service







FEDERATE SECURELY

OTHER ORGANIZATION

Once indexing metadata is applied indicating that a collection should be indexed, a job is submitted to the iRODS delayed execution queue which will perform the requested action asynchronously.

Data Virtualization ( Unified Namespace )

Data Discovery ( Metadata )

Workflow Automation ( Rule Engine )

Secure Collaboration ( Federation )

# Tagging collections for indexing

Collections are tagged with metadata to indicate they should be indexed

A new AVU applied to a populated collection will schedule all objects for indexing

New objects placed into a collection with one or more indexing AVUs applied will also be indexed

# Tagging collections for indexing

Objects that are modified or moved into a collection with one or more indexing AVUs applied will also be indexed

Indexing policy is inherited from parent collections:

a parent collection indexing metadata is also applied to any sub-collections

# Tagging collections for indexing

Indexing metadata takes the form:

A: irods::indexing::index
V: <index name>::<index type>
U: <technology>

- index name is specific to your index configuration
- index type is either: **full_text** or **metadata**
- technology specifies which policy will be invoked to perform the indexing - currently elasticsearch

# Configuring Indexing Resources

An administrator may wish to restrict indexing activities to particular resources, for example when automatically ingesting data.

In order to indicate a resource is available for indexing it may be annotated with metadata:

```
imeta add -R <resource name> irods::indexing::index true
```

If no resource be tagged it is assumed that all resources are available for indexing.

**Should the tag exist on any resource in the system, it is assumed that all available resources for indexing are tagged.**

# Overriding the Indexing Policy

Policy Signatures - Implement these four policies to provide service to a new technology

**irods_policy_indexing_object_index_<*technology*>**(
*object_path, *source_resource, *index_name, *index_type)
**irods_policy_indexing_object_purge_<*technology*>**(
*object_path, *source_resource, *index_name, *index_type)
**irods_policy_indexing_metadata_index_<*technology*>**(
*object_path, *attribute, *value, *unit, *index_name)
**irods_policy_indexing_metadata_purge_<*technology*>**(
*object_path, *attribute, *value, *unit, *index_name)

# Indexing Policy

The Indexing Policy provides a reactive framework to metadata attributes.  Once the indexing technology policy is invoked, it may provide any implementation desired.

For instance, given a document type, a Solr implementation can implement geographic indexing rather than full text for the "full_text" type and ignore the "metadata" type.

An implementation for Jena would ignore the "full_text" type and only implement the metadata policies.

A policy framework that provides an asynchronous, scalable data publishing service driven by metadata

- Publishing technology of choice is reached by delegating policy implementation
- Persistent identifier generation is delegated to a policy invocation

# Publishing Policy Components

- Persistent Identifier
- Publishing Policy Implementation
  - irods_policy_publishing_object_publish_<technology>
  - irods_policy_publishing_object_purge_<technology>
  - irods_policy_publishing_collection_publish_<technology>
  - irods_policy_publishing_collection_purge_<technology>

<technology> is directly derived from metadata and is used to delegate the policy invocation

# Publishing Overview

iRODS

The iRODS Publishing Capability provides a metadata driven policy framework for the implementation of data publication to external services.

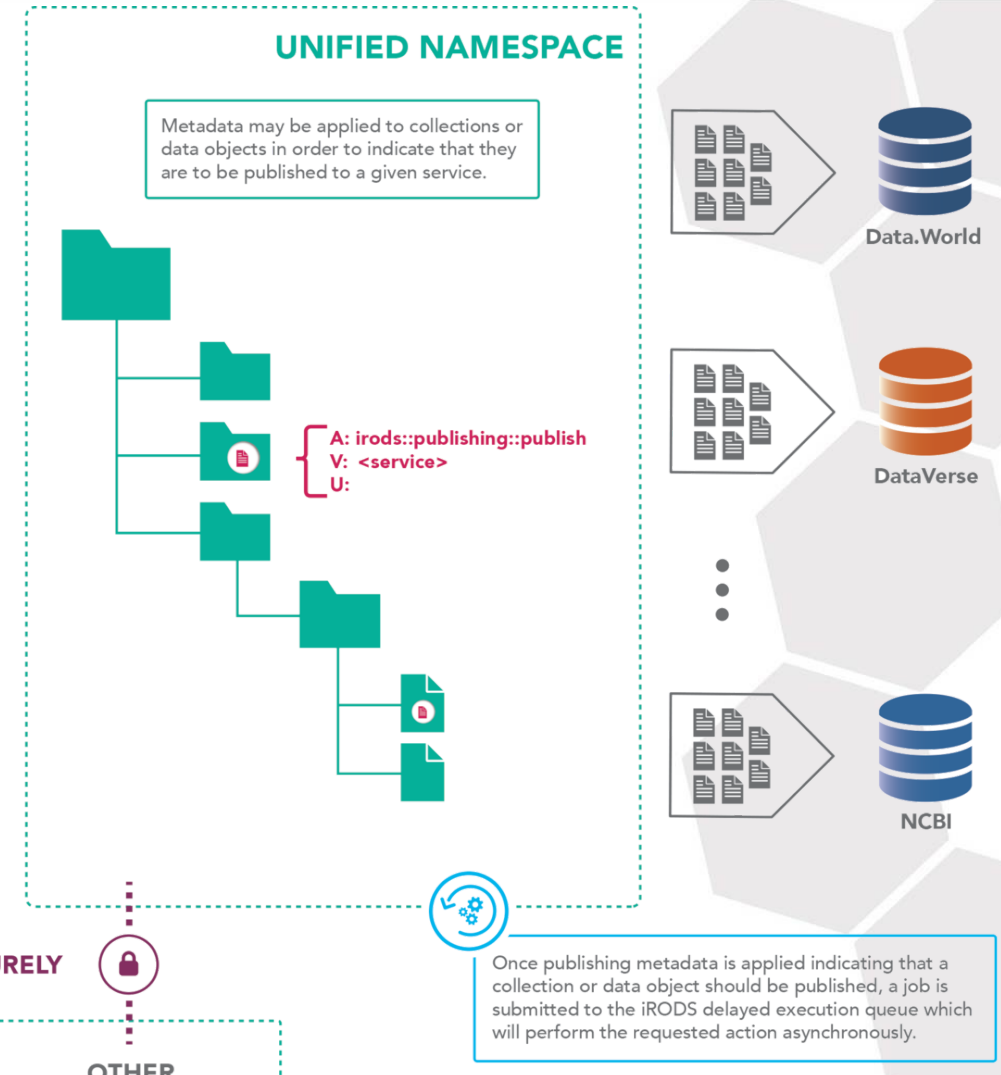
The policy framework provides:

- Protection for published data, which prevents future modification
- Invocation of secondary policy for the generation and application of persistent identifiers
- Advertisement and possible movement of published data to external catalogs

From the configured metadata, the framework composes a rule name and then delegates to the policy implementation through the rule engine.

A new publishing service can be supported via a rule base or policy engine which provides policy implementations of the form:

- irods_policy_publishing_object_publish_<service>
- irods_policy_publishing_object_purge_<service>
- irods_policy_publishing_collection_publish_<service>
- irods_policy_publishing_collection_purge_<service>




Data Virtualization ( Unified Namespace )

Data Discovery ( Metadata )

Workflow Automation ( Rule Engine )

Secure Collaboration ( Federation )

# Tagging collections for publishing

Collections and Data Objects are tagged with metadata to indicate they should be published

A new AVU applied to a populated collection will schedule all objects for publication

New objects cannot be placed into a collection with a publishing AVUs applied.  Nor can those objects be modified with POSIX operations.

# Tagging for publication

Publishing metadata takes the form:

A: irods::publishing::publish
V: <service>

The service name is directly applied the the policy name template, which dictates which policies are invoked.

# Immutability of Published Content

## Users cannot modify or delete published content

```
irm -f published_file0
```

```
remote addresses: 127.0.0.1 ERROR: rmUtil: rm error for
/tempZone/home/irodsconsortium/published_file0, status = -35000
status = -35000 SYS_INVALID_OPR_TYPE
Level 0: object is published and now immutable
[/tempZone/home/irodsconsortium/file3]
```

## Users cannot remove publication metadata

```
imeta rm -d file3 irods::publishing::publish dataworld
```

```
remote addresses: 127.0.0.1 ERROR: Level 0: publishing metadata tags
are immutable [/tempZone/home/irodsconsortium/file3]
remote addresses: 127.0.0.1 ERROR: rcModAVUMetadata failed with
error -35000 SYS_INVALID_OPR_TYPE
Level 0: publishing metadata tags are immutable
[/tempZone/home/irodsconsortium/file3]
```

# Overriding the Publishing Policy

Policy Signatures - Implement these four policies to provide integration to a new publishing service

**irods_policy_publishing_object_publish_<service>**(
*object_path, *user_name, *service_name)

**irods_policy_publishing_object_purge_<service>**(
*object_path, *user_name, *service_name)

**irods_policy_publishing_collection_index_<service>**(
*collection_name, *user_name, *service_name)

**irods_policy_publishing_collection_purge_<service>**(
*collection_name, *user_name, *service_name

# Publishing Policy

The Publishing Policy provides a reactive framework to metadata attributes.  Once the publishing service policy is invoked, it may provide any implementation desired.

For instance, some services may simply need a URI to the data set whereas others may require the data be uploaded, such as data.world.

The publishing service may require a specific submission package format, additional metadata or other requirements which would require the publishing job to wait until these needs are met.

# Future Work - New services to support

Indexing

- Solr - geographic indexing
- Semantic indexing technologies
- Tika data typing

Publishing

- Dataverse
- Life science catalogs
- Handle
- DOI
- Minid

This should be a community discussion

# Questions?