Policy-Driven Repository Interoperability: Enabling Integration Patterns for iRODS and Fedora

David Pcolar
Carolina Digital Repository (CDR)
UNC Chapel Hill
david_pcolar@unc.edu

Daniel W. Davis
Cornell Information Sciences (CIS)
DuraSpace Affiliate
dwdavis@cs.cornell.edu

Bing Zhu
Data Intensive Cyber Environments (DICE)
University of California: San Diego
bizhu@ucsd.edu

Alexandra Chassanoff
School of Information & Library Science (SILS)
UNC Chapel Hill
achass@email.unc.edu

Chien-Yi Hou
Sustainable Archives & Leveraging Technologies (SALT)
UNC Chapel Hill
chienyi@unc.edu

Richard Marciano
Sustainable Archives & Leveraging Technologies (SALT)
UNC Chapel Hill
richard_marciano@unc.edu

ABSTRACT

Given the growing need for cross-repository integration to enable a trusted, scalable, open and distributed content infrastructure, this paper introduces the Policy-Driven Repository Interoperability (PoDRI) project investigating interoperability mechanisms between repositories at the policy level. Simply moving digital content from one repository to another may not capture the essential management policies needed to ensure its integrity and authenticity. Platform-independent, policy-aware object models, including policy expressions, and a distributed architecture for policy-driven management are fundamental building blocks of a sustainable access and preservation infrastructure. This project integrates iRODS and Fedora to demonstrate such an infrastructure. Using iRODS and its rules engine, combined with Fedora's rich semantic object model for digital objects, provides the basis for implementing a policy-driven test-bed. Using a policy-driven architecture is an essential part of realizing a fully model-driven repository infrastructure capable of decoupling the permanent digital content from the constantly evolving information technology used to support them.

1. INTRODUCTION

This paper introduces the Policy-Driven Repository Interoperability (PoDRI) project investigating interoperability between repositories at the policy level. PoDRI is led by the University of North Carolina at UNC, with units ranging from SALT (Sustainable Archives & Leveraging Technologies), RENCI (Renaissance Computing Institute), SILS (School of Information and Library Science), and the Libraries/CDR (Carolina Digital Repository). Key partners include Bing Zhu at UCSD (DICE, Data Intensive Cyber Environments) and Daniel Davis at DuraSpace (combining DSpace and Fedora Commons) and Cornell Information Sciences. The project is sponsored by an IMLS National Leadership grant and is motivated by the growing need to create a scalable, open and distributed infrastructure that provides durable, trusted access and management of our valuable digital content of all kinds (e.g. research data sets, documents, video, metadata). This problem is well described in the NSF's Cyberinfrastructure Vision for the 21st Century [14].

Simply replicating digital content from one repository, with or without any associated metadata, may not capture the essential management policies that ensure integrity and authenticity, a critical requirement for establishing a trust model. "A policy is typically a rule describing the interactions of actions that take place within the archive, or a constraint determining when and by whom an action may be taken [8]." Typical policies include those that control data ingestion, administration, preservation, access procedure, authentication and authorization.

A distributed policy management architecture is an essential component in realizing a trust mechanism for repository interoperability. The PoDRI project investigates the requirements for policy-aware interoperability and demonstrates key features needed for its implementation. The project is focused on integrating object models, including interoperable policy expressions, and a policy-aware distributed architecture that includes both repositories and middleware services.

Our overarching design paradigm is that permanent digital content must be decoupled from the constantly evolving infrastructure supporting it. Increasingly, the information infrastructure will be a part of a global, interoperable, heterogeneous, distributed "system-of-systems". Model-driven methods will be essential to make the management of such an infrastructure feasible; policy-driven methods are a core, enabling part of governance mechanisms needed to ensure control and preservation of our permanent digital content.

The PoDRI project addresses the following research problem: **What is the feasibility of repository interoperability at the policy level?** Research questions to be addressed are:

- Can a preservation environment be assembled from two or more existing repositories?
- Can the policies of the federation be enforced across repositories?
- Can policies be migrated between repositories?
- What fundamental mechanisms are needed within a repository to implement new policies?

iRODS (integrated Rule-Oriented Data System) [12,14] and the Fedora Repository [7,9] will be used as representative open source software to demonstrate the

PoDRI architecture. Combining iRODS and Fedora enables use of the best features of both products for building sustainable digital repositories. iRODS provides an integrated rule engine, distributed virtual storage, the iCAT[1], and micro-services[2]. Fedora offers rich semantic object modeling for digital objects, extensible format-neutral metadata and a flexible service mediation mechanism.

2. RATIONAL FOR IRODS-FEDORA INTEGRATION

Early in 2006, the DART project [3] created an SRB storage interface for Fedora that allows all Fedora digital content, including Fedora Digital Objects (FDO) and their Datastreams, to be stored in SRB distributed repositories. Similarly, a storage module was developed by Aschenbrenner and Zhu [1] for iRODS. Using the Fedora-iRODS storage module, iRODS can act as a back-end for Fedora and, thus, provide opportunities for Fedora to use iRODS capabilities such as virtual federated storage, micro-services and the rules engine.

iRODS offers an appealing platform for implementing a distributed policy-driven management architecture. The integrated rules engine can be used to invoke a range of rules including policy expressions and, through the use of micro-services, can execute code for those policies in a distributed environment. Rules can act as simple workflows performing a sequence of pre-defined actions. iRODS rules can be executed explicitly, triggered by external conditions or events and executed at timed intervals. For example, iRODS can implement a replication policy, geographically disbursing file copies across the network. Micro-services can be written for feature extraction, format migration, integrity checks and other preservation services.

While used to efficiently hold and query structured data and metadata, the iCAT relational database is not optimal for handling the complex, variable metadata needed for preservation and curation. Indeed, any relational database will require considerable coding to support complex metadata schemas, making the use of unstructured data (files) possibly in combination with XML databases or semantic triplestores as a more flexible alternative [10].

Fedora is file-centric; all Fedora data and metadata is stored in files [6]. The Fedora Digital Object (FDO), a kind of compound digital object, provides the organizing metadata used to "make sense" of itself and other resources. It uses the FOXML schema to encapsulate metadata, and to reference other files or web resources. Since the FDO is a file, it can be stored in iRODS like any other file.

Digital content (or user-defined metadata) managed by the FDO is stored in one or more separate files — each registered in a FOXML element called a Datastream. Datastreams can also capture relationships to other objects and external resources. Users may add metadata to the FDO or add additional metadata Datastreams (to be stored like any other file).

This means, however, that metadata is stored in an unstructured format, often XML or RDF, and requires external indices to support querying by search engines, semantic triplestores, XML databases, and the iCAT. Fedora's approach provides a format neutral, extensible framework for representing data and metadata.

The rich metadata environment provided by the FDO can augment the structured metadata found in the iCAT. Metadata can be copied from the iCAT into a more easily preserved unstructured file format, as demonstrated by Bing Zhu and colleagues [17]. Critical data can be copied from the FDO, or as user metadata files (Datastreams), so they can be queried from the iCAT. With suitable metadata both the iCAT and the Fedora repository could be entirely rebuilt from files if the indices were lost or corrupted.

Fedora has a set of "front-end" APIs that provide the means to ingest and manipulate FDOs (CRUD). iRODS is capable of calling these APIs to perform operations from micro-services. Fedora also provides an extensible mechanism to add custom functionality called "services" that are executed within the context of the FDO. Services act as extensions to the "front-end" API of the object. Fedora mediates the service request calling the appropriate "back-end" functionality. The back-end functionality can be a Web service, in this case potentially provided by iRODS. Custom Fedora services provide another mechanism to interact with iRODS. Since iRODS can interact with Fedora's "front-end" APIs, "back-end" services, and the Fedora-iRODS storage module, one may picture iRODS wrapping around Fedora.

3. ENABLING A POLICY-DRIVEN MANAGEMENT ARCHITECTURE

To demonstrate distributed policy-driven management architecture, we plan to implement the following operational scenarios:

- Integrate views of content, original arrangement (hierarchy) and metadata
- Create an audit trail of policy execution events and related provenance information
- Manage policies through Fedora
- Show iRODS invoking policies from Fedora

Both iRODS and Fedora fully support distributed computing installations. In effect, both products can be characterized as virtualization middleware for storage, access and service execution. The products, however, have very different operational paradigms which must be accommodated but provide complementary strengths that can be exploited when used together.

[1] iCAT is the metadata catalog in iRODS that stores metadata about all objects in iRODS in relational databases.

[2] Microservices are function snippets or executables that can be used to perform a distinct task using well-defined input information structures.

The virtual file system in iRODS makes it the logical choice for all storage (including FDOs). In addition, the iRODS rules engine and micro-services provide an effective means for orchestrating services such as policy invocation. Fedora's capabilities, on the other hand, are especially powerful for handling variable content and different metadata formats, for flexibly relating resources, facilitating presentation (manifestation of content), and its mediation capabilities make it appealing in building systems that are "designed for change."

A policy-driven management architecture requires that policy expressions be persistent. Fedora could be used to create FDOs containing policy expressions, which would subsequently be loaded into machine-actionable form and invoked as iRODS rules. Since policies are part of an object's provenance, Fedora can relate the policy FDOs to content items in which they apply. Because policy invocation will be performed by iRODS, audit records of the execution must also be created by iRODS. Subsequently, iRODS will store the execution records back into Fedora as FDOs, linking them to the FDOs containing the content and policy expressions.

iRODS does not currently generate audit data in a format compliant with the PREMIS preservation metadata schema. The CDR, however, implements auditing of objects via a PREMIS.XML file for each iRODS data object. This method may not be sustainable for repositories containing millions of objects. Preservation activities, such as replication or fixity checks, generate large amounts of log entries over time and potentially exceed the byte size of the original object. Discussions between CDR and iRODS developers suggest multiple methods for retaining and aggregating various component logs for translation into PREMIS-compliant events. Do we continue to store these events with the individual objects or as an aggregate? Do we generate specific PREMIS information upon request? In the case of replicas residing on disparate nodes in a data grid, auditable events will occur that differ from those affecting the original object. How do we reconcile these events in a singular view of the object?

Users and user applications will still need to interact with Fedora or iRODS directly. This is particularly true of research (grid) applications with large datasets. Select metadata will need to be duplicated in both products to access content, to represent relationships, and to preserve integrity and authenticity. Direct interaction by users or user applications with either Fedora or iRODS will require both products to synchronize or update metadata.

These interactions may trigger policy invocations. For example, Fedora may trigger policy invocation indirectly when interacting with a file (CRUD) or directly through a Fedora custom service. Conversely, iRODS' micro-services can call Fedora services to provide feedback in the system.

A more comprehensive "Concept of Operations" document will be prepared as part of the PoDRI project. The following set of questions is drawn from our current understanding of the operational scenarios:

- How will the collection structure be represented in the two products?
- How will Fedora be initialized for existing content in iRODS?
- How will Fedora be informed of content or metadata changes initiated directly in iRODS?
- How can content or metadata from Fedora be accessed by iRODS services?

4. ENABLING USE CASES

Five enabling use cases have been identified for the Fedora-iRODS integration. These use cases are:

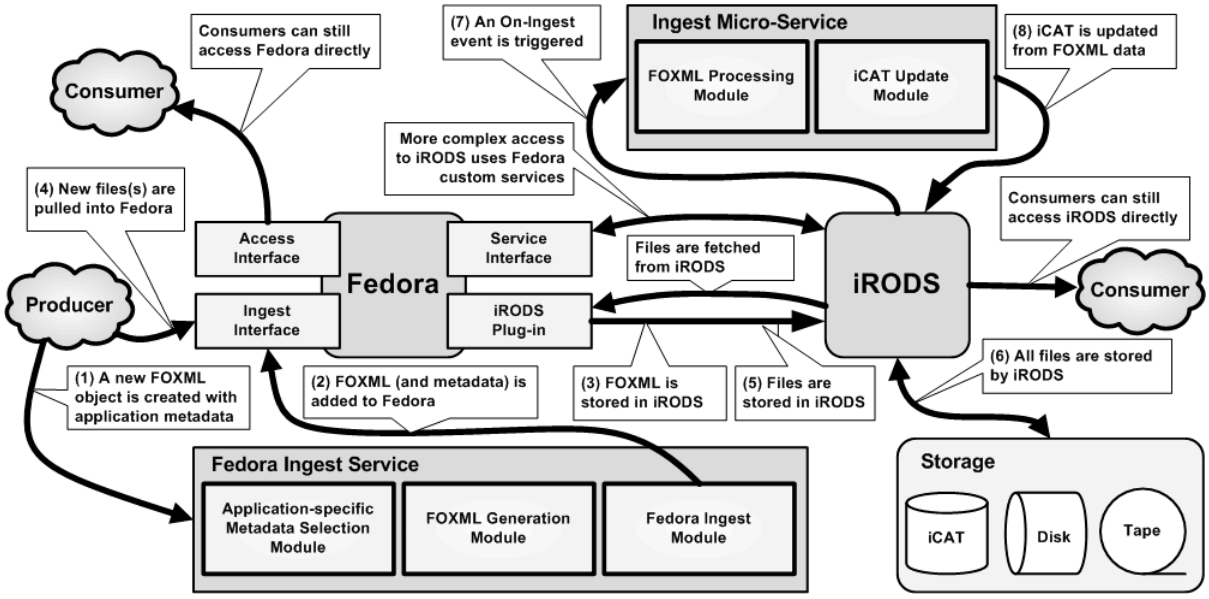

Figure 1: New Content Ingest via Fedora

1. New content ingest via Fedora
2. New content ingest via iRODS
3. Bulk registration from iRODS into Fedora
4. Update of content or metadata via Fedora
5. Update of content or metadata via iRODS

We introduce each of these use cases in this paper. While they do not by themselves represent policy management operations, they are prerequisites for enabling policy-driven operations and represent *demonstrations* of policy interoperability between repositories. The initial implementation work is focused on uses cases one and two together with the storage plug-in, a key enabler, described in Section 5.1.

4.1. New Content Ingest via Fedora

Current users of Fedora will want to continue ingesting into Fedora. Users are also likely to use Fedora features to add and relate rich metadata including policy, provenance and authenticity information. As shown in Figure 1, when new content is ingested into Fedora, it is able to capture the metadata needed for its operation. Digital content (or user-defined metadata) is either pulled in by Fedora or pushed to Fedora and stored in individual files. The file containing the FDO (FOXML) and the content files are subsequently stored in iRODS with no permanent storage directly managed by Fedora.

Selected metadata is collected by Fedora during the ingest process and stored in an internal system index implemented using a relational database. This database is used only to speed up access to content or bindings to services (formerly called disseminators). Optionally, metadata or notifications can be sent to index services such as semantic triplestores, search engines and OAI-PMH harvesters.

The Carolina Digital Repository (CDR) is using Solr/Lucene as the indexing and search engine for discovery of ingested content. Metadata is extracted during the ingest process from MODS and FOXML files.

Objects ingested via Fedora and stored in iRODS do not, by default, retain the logical tree structure of the original file system. Instead, CDR preserves the hierarchal structure of the file system via relations in the RDF triple store.

The arrangement of objects is achieved by creating FDOs representing the parent and child. The relationship is recorded in RDF (within the RELS-EXT Datastream) using the "isMemberOf" relation asserted in the child to the parent. The obverse relation "hasMember" is implied and could be stated explicitly in the parent. These two relations provide a way to build a hierarchical structure for all objects, collections and files. In Fedora, these relations form a "graph" and objects may participate in any number of graphs using other relations and, therefore, are not limited to a single hierarchy. Relationship information can be accessed by introspecting on the FDO or the relations can be indexed into a RDF triplestore [16] and queried by applications to extract a graph for navigating from parent to children as people usually do for a tree structure. Similar methods can be used to navigate any relationship graph.

How will the metadata in iRODS be updated in this use case? Two alternatives being considered are: (1) call a Fedora custom service to update the iCAT; (2) when the FOXML file is ingested, a monitoring rule can trigger an iRODS micro-service to introspect on the FDO to extract the metadata.

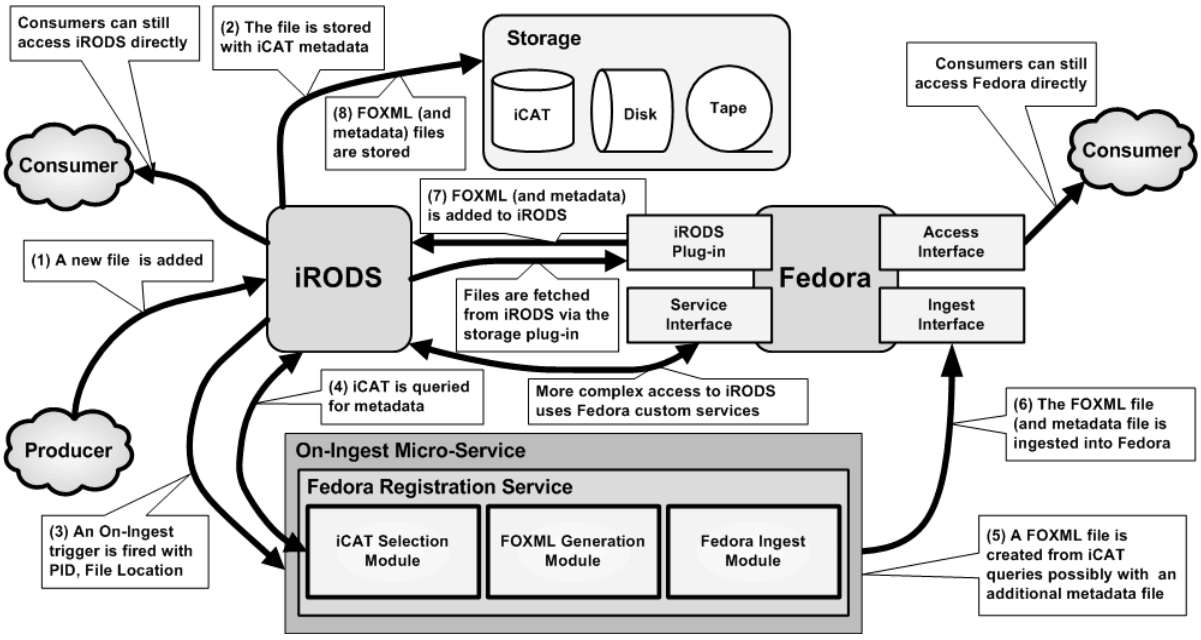

Figure 2: New Content Ingest via iRODS

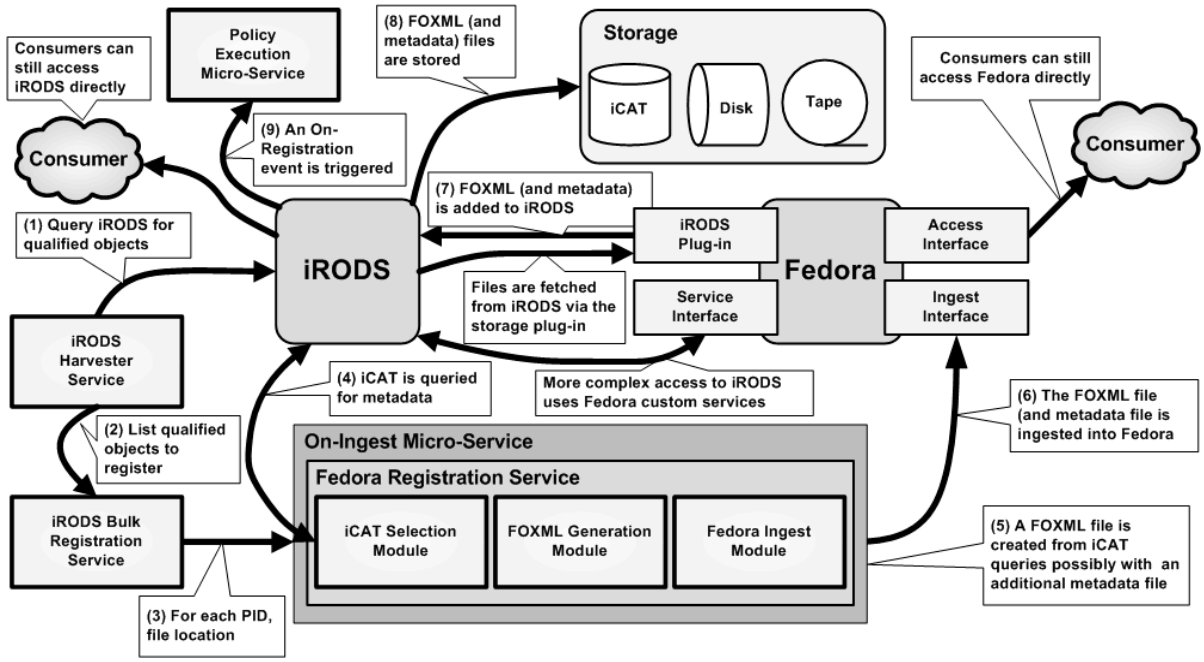

Figure 3: Bulk Registration into Fedora

4.2. New Content Ingest via iRODS

Current iRODS users will likely want to continue to use iRODS directly to store data objects, particularly in research settings where direct access to storage is desired. The digital content (data object) is typically ingested into iRODS as a file operation. In iRODS, the hierarchical relation of a data object and its ancestors are encoded and described explicitly in its global object name. Two questions arise from this scenario. First, how will Fedora be notified of arrival of the new data object? Second, how will an analog component to its iRODS hierarchy be represented in Fedora?

As depicted in Figure 2, a utility is needed to register iRODS files into Fedora. A micro-service could call this utility when triggered by a monitoring rule on the storage operation which would create the FDO for the data object and ingest it into Fedora. The micro-service can be deployed as a rule under the iRODS rule event, 'acPostProcForPut'. Once this rule is activated in an iRODS server, the micro-service can be triggered after each new iRODS data object is created in a specified collection in the iRODS Content Store (see iRODS Storage Module). It will create pre-ingest FOXML for the new data object, querying the iCAT for additional metadata as needed. Within the FOXML, it will create a Datastream containing a reference to the location of the data object within iRODS. It will then ingest the FOXML using Fedora's API-M to create the FDO. This rule is activated once placed in the rule configuration file of an iRODS server. It will monitor all file activities in the iCAT catalog and will create an FDO for any newly created iRODS file.

When using iRODS for back-end storage, all FDOs and Datastreams are stored in iRODS as files in one of two collections: FOXML Object Store and iRODS Content Store. Therefore, users can directly access the

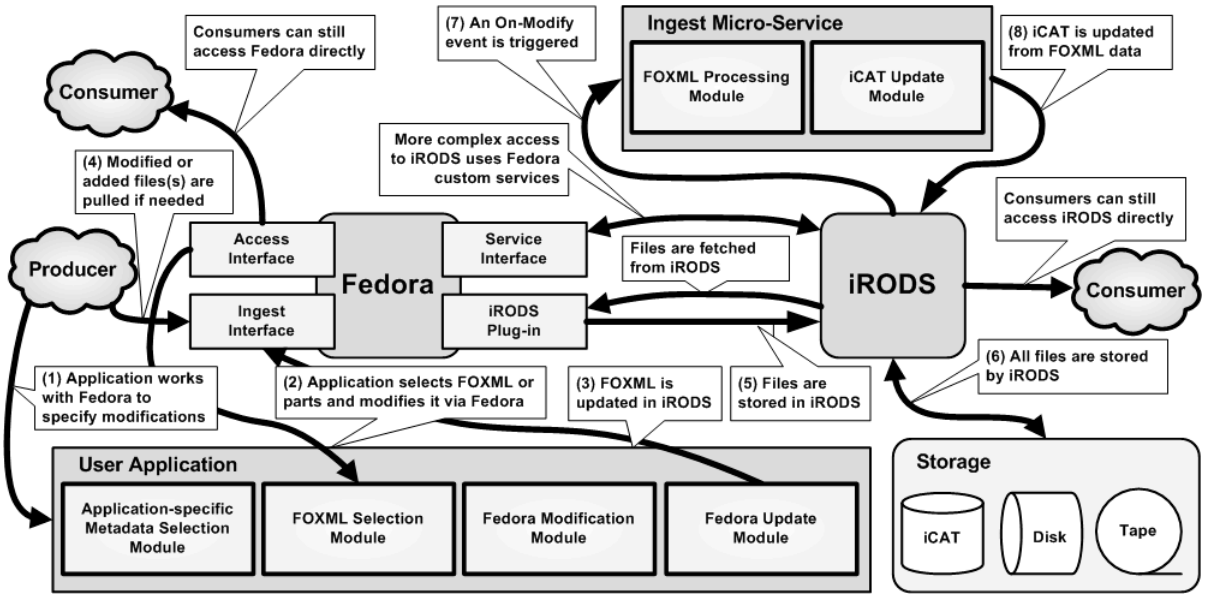

Figure 4: Update Content or Metadata via Fedora

files containing Fedora metadata through the iRODS interface. On the other hand, files stored in iRODS, whether for an FDO or a Datastream, have both an independent set of iRODS system metadata as well as a set of user-defined metadata. The system metadata contains important information for each replica of an iRODS file, including the file's location, storage type, audit trail, and associated iRODS rules. The two sets of metadata can be represented as external Datastreams in FOXML and generated dynamically when accessed using the Fedora-iRODS storage module.

As described above, Fedora uses RDF relations to describe the arrangement of objects. This requires the creation of FDOs representing each hierarchical level which has the advantage of enabling the participation of iRODS in the semantic network functionality provided by Fedora. Since iRODS can create a virtual hierarchy, it may not be desirable to instantiate corresponding FDOs. Users can create custom Datastreams as "finding aids"; the virtual hierarchy can then be encoded using RDF or any other desired format. Similar to iRODS, parent-child relationships can be modeled as path metadata and stored in the custom Datastream. An application or a Fedora custom service can be used to interpret the format of the Datastream to display the hierarchy [5].

One of the CDR's core constituencies are the special collections in our university libraries. These collections tend to have rich metadata associated with them and have usually undergone preliminary curation. The longer term goal of the repository is to harvest content directly from research-based iRODS data grids. Metadata quality and quantity is typically limited in these collections. Repository outreach and development is concerned not only with identifying and preserving "at risk" collections, but cultivating metadata collection and data curation proactively throughout the research lifecycle.

4.3. Bulk registration from iRODS into Fedora

Often we will be presented with existing collections in iRODS which we want to add to Fedora. How will these collections be registered into Fedora? It would be time consuming to require manual extraction, encapsulation (in an FDO) and storage of each data object.

As shown in Figure 3, a four step process is needed to automate this process: (1) identify the iRODS data objects to register; (2) iterate over each data object; (3) automatically collect metadata about each data object; (4) create the analogous FDO and ingest it via Fedora (possibly with additional FDOs to represent the hierarchy).

Bulk registration of a collection of iRODS files could be deployed and executed by a data curator through a single command irule, an iRODS command to send and execute a rule in an iRODS server. Often, such a rule is executed for a collection recursively. Registering multiple collections can be accomplished by through a batch script, which could query all iRODS files within a specified collection and create an FDO for each iRODS file. All FDOs could then be stored back into iRODS in the FOXML Object Store. Note that executing this process through iRODS facilitates inclusion of feature extraction services to automate metadata extraction. Also, note that services often can be reused in multiple use cases, reducing software development and deployment costs.

4.4. Update Content or Metadata via Fedora

Updates via Fedora use the same techniques as when iRODS is not present (see Figure 4). A user or application uses the Fedora APIs to update metadata, add new Datastreams, or new Datastream versions. Note that when a Datastream is versioned, it will result in a new file in iRODS.

However, iRODS must update system and user metadata in the iCAT by interpreting the updated FDO, extracting modified metadata, and updating the iCAT.

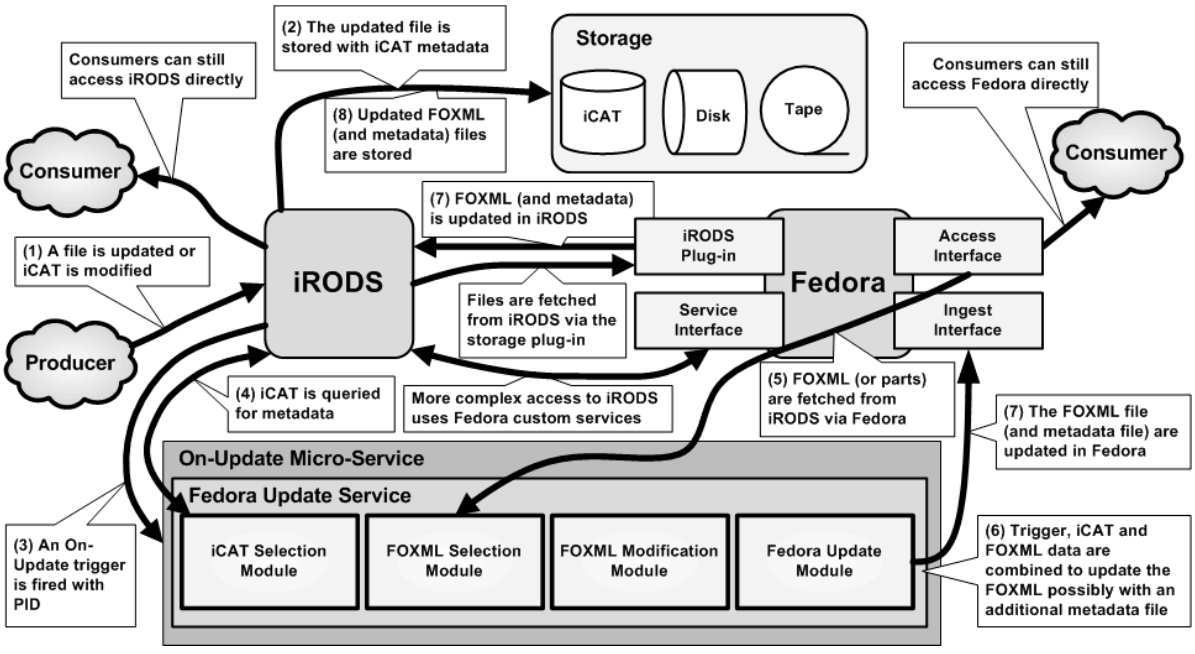

Figure 5: Update Content or Metadata via iRODS

This operation is triggered when the modified FDO is saved causing the execution of a micro-service to perform these tasks. The micro-service will then be able to query the iCAT to fetch data that will help and update the iCAT.

4.5. Update Content or Metadata via iRODS

An update via iRODS is similar to registering a new iRODS file into Fedora. As shown in Figure 5, however, the FDO already exists and must first be fetched, revised and updated in Fedora. The Fedora APIs are also capable of performing more fine-grained operations on the FDO for very common updates. For example, if a new file is added to iRODS, it may be registered in a new FDO (see *Ingest New Content via iRODS*). However, since Fedora supports a compound object model, the new file could be added as a new Datastream to an existing FDO. In this case, the addDatastream API method is preferred. Similar convenience functions exist for updating relations and certain metadata elements.

If a file is updated in iRODS, metadata (for example, the "last update" timestamp) for the Datastream in Fedora must also be updated. If a new version of a file is added to iRODS, the updateDatastream API method is used. This can only be used if the new version is represented by a new file in iRODS.

The update case puts a significant burden on the micro-service to determine the best approach to update Fedora. In particular, the micro-service must be informed of the FDO which correlates with the iRODS file. The iCAT will have to be extended to include the Fedora PID.

5. ADDITIONAL UTILITIES

We are implementing two key enabling utilities in addition to the functionality described above. First is an updated storage module as an iRODS-specific plug-in to replace Fedora's Low-level Store. Second is a harvester utility which can be used in both bulk registration and for disaster recovery.

5.1. iRODS Storage Module

We plan to store all files in iRODS. This will require an update of the existing iRODS-Fedora Storage Module or building a new one. Because this is a key enabler, work is concentrating on updating the existing iRODS plug-in replacing the Fedora Low-Level storage module. A new storage module is also being built, using Jargon and the Fedora Commons Akubra interface. Furthermore, a storage module is being developed by Aschenbrenner, based on the Merritt Storage System [2], in an iRODS community project. We are also closely following work in DuraCloud for integrating with cloud storage and service providers [13]. Selecting these candidates was based on a survey of available storage subsystems, finding a great proliferation of new approaches. Testing, however, eliminated all FUSE-based solutions as too unreliable except for the most lightweight usage. Building a new storage module, based on one or more of these existing technologies, would permit research on using it as a feedback path for policy operations including security policies.

When iRODS serves as a storage module for Fedora, the current design is to use two iRODS collections: (1) Fedora Digital Objects (FOXML) in the FOXML Object Store, and (2) content objects (Datastreams) in the iRODS Content Store. They are accessed through a single curator user account in iRODS. This makes it easier to distinguish between policies related to FDOs from those operating on content objects (Datastreams).

This approach, however, differs from the Fedora/Jargon/Merritt default of storing objects in folders based on a directory/file path and naming scheme. For the CDR and other existing implementations, a restructuring of objects into the segregated object store will be required. This will alter iRODS based failure recovery mechanisms and integrity audits.

5.2. iRODS Data Harvester for Fedora

The iRODS Data Harvester is an adaptive version of the Data Rebuilder in Fedora. It is used to re-build the object indices from the FOXML Object Store and iRODS Content Store. It does not create any new FOXML objects; rather, it surveys all the objects stored within the FOXML Object Store, verifies the Datastreams inside the iRODS Content Store, and creates the indices in the database used by the Fedora server. The iRODS Data Harvester also builds the necessary RDF data to be stored in the RDF triplestore for the navigation of hierarchical structure.

6. POLICY FEDERATION AND MIGRATION

The iRODS rule engine provides the capability to apply rules on the data grid side to implement the policies. The Distributed Custodial Archival Preservation Environments (DCAPE) project [4] aims to work with a group of archivists to develop a set of rules to automate many of the administrative tasks associated with the management of archival repositories and validation of their trustworthiness. These DCAPE rules could be applied to different repositories based on the institution's policies. We plan to provide the functionality for users to manage the policies through the Fedora interface and be able to check what rules are in action.

Current implementations, even in data grid environments, depend on local enforcement of policies and typically do not consider the larger framework of uniform policy implementation across heterogeneous repositories. Though currently still in development, the ISO/NP 1636 standard [11] could present a model for identification of machine-actionable rules that can be expressed as policies. Stored as Fedora Service Definitions, the policies will have unique service deployment bindings for each data storage system. Our demonstration storage implementation is iRODS, but

other storage environments may be supported by changing deployment mechanisms.

The CDR is developing a policy management framework based on a machine interpretable series of actions across repositories in a data grid. Implementation of new policy requires identification of machine-actionable components and mapping to specific, testable deployment mechanisms.

7. SUMMARY

In this paper, we introduced the Policy-Driven Repository Interoperability (PoDRI) project investigating interoperability mechanisms between repositories at the policy level. The rationale for using iRODS and Fedora to demonstrate key features of a distributed policy-driven management architecture was described. Four scenarios that will be demonstrated as part of the project were enumerated. We have identified five enabling use cases that are needed for the demonstration scenarios along with two key utilities planned for development. We also introduced work on policy federation and migration. PoDRI is an applied research project and its details will change as we develop a greater understanding of the methods for policy-driven interoperability.

8. ACKNOWLEDGEMENTS

This project is funded by IMLS grant LG-06-09-0184-09 as part of the 2009 National Leadership Grants NLG Library-Research and Demonstration, awarded to the University of North Carolina at Chapel Hill. Project Director is Richard Marciano. Collaborators at UNC / SILS include: Alex Chassanoff, Chien-Yi Hou, Reagan Moore, and Helen Tibbo. At UNC / Libraries: Steve Barr, Greg Jansen, Will Owen, and Dave Pcolar. At UNC / RENCI: Leesa Brieger. At UCSD: Bing Zhu. At DuraSpace and Cornell Information Sciences: Daniel Davis and Sandy Payette. Finally, at the University of Maryland iSchool: Bruce Ambacher.

9. REFERENCES

[1] Aschenbrenner, A., Zhu, B. "iRODS-Fedora Integration", http://www.irods.org/index.php/Fedora

[2] California Digital Library. "Merritt storage service", http://confluence.ucop.edu/display/curation/storage

[3] DART, University of Queensland. "Fedora-SRB Database integration module", http://www.itee.uq.edu.au/~eresearch/projects/dart/outcomes/FedoraDB.php

[4] DCAPE. "Distributed custodial archival preservation environments", an NHPRC-funded project, http://dcape.org

[5] DuraSpace. "The Content Model Architecture", http://fedora-commons.org/confluence/x/gABI

[6] DuraSpace. "The Fedora Digital Object Model", http://fedora-commons.org/confluence/x/dgBI

[7] DuraSpace. "Fedora Repository 3.3 documentation", http://fedora-commons.org/confluence/x/AgAU

[8] DuraSpace. "PLEDGE project", http://fedora-commons.org/confluence/x/WSDS

[9] Fedora Commons, http://www.fedora-commons.org

[10] Hedges, M., Hasan, A., and Blanke, T. "Management and preservation of research data with iRODS", *Proceedings of the ACM first workshop on CyberInfrastructure:Information management in eScience*, Lisbon, Portugal, pp. 17-22, 2007. doi: http://doi.acm.org/10.1145/1317353.1317358

[11] International Organization for Standardization. "ISO/NP 16363: Audit and certification of trustworthy repositories", http://www.iso.org/iso/catalogue_detail.htm?csnumber=56510

[12] iRODS: Data grids, Digital Libraries, Persistent Archives, and Real-time Data Systems. http://www.irods.org

[13] The Library of Congress. "DuraCloud", http://www.digitalpreservation.gov/partners/duracloud/duracloud.html

[14] Moore, R., Rajasekar, A., Wan, M., and Schroeder, W. "Policy-based distributed data management systems", *The 4th International Conference on Open Repositories*, Atlanta, Georgia, May 19, 2009.

[15] NSF Cyberinfrastructure Council. "NSF's cyberinfrastructure vision for 21st century discovery", National Science Foundation, March 2007, http://www.nsf.gov/pubs/2007/nsf0728/index.jsp?org=NSF

[16] Wikipedia. "Triplestore", http://en.wikipedia.org/wiki/Triplestore

[17] Zhu, B., Marciano, R., and Moore, R. "Enabling Inter-Repository Access Management between iRODS and Fedora", *The 4th International Conference on Open Repositories*, Atlanta, Georgia, May 19, 2009