# *Digital repository: preservation environment and policy implementation*

**Bing Zhu, Richard Marciano, Reagan Moore, Laurin Herr & Jurgen Schulze**

**International Journal on Digital Libraries**

ISSN 1432-5012

Int J Digit Libr
DOI 10.1007/s00799-012-0082-3


Springer

Int J Digit Libr
DOI 10.1007/s00799-012-0082-3

# Digital repository: preservation environment and policy implementation

**Bing Zhu · Richard Marciano · Reagan Moore · Laurin Herr · Jurgen Schulze**



**Abstract** Distributed digital repositories can be used to address critical issues of long-term digital preservation and disaster management for large data centers. A policy-driven system provides an ideal solution for managing distributed repositories that require high flexibility and high configurability. Recent studies demonstrate that the integrated Rule-Oriented Data System, a peer-to-peer server middleware, provides the requisite dynamic extensibility needed to manage time-varying policies, automate validation of assessment criteria, manage ingestion processes, manage access policies, and manage preservation policies. The policy management can be implemented underneath existing digital library infrastructure such as Fedora.

**Keywords** Digital preservation · Policy-driven system · Policy enforcement · Micro-service oriented · Reconfigurable and automated services

B. Zhu (✉)
San Diego State University, San Diego, CA, USA
e-mail: zhu@sciences.sdsu.edu

R. Marciano · R. Moore
University of North Carolina, Chapel Hill, Chapel Hill, NC, USA
e-mail: richard_marciano@unc.edu

R. Moore
e-mail: rwmoore@renci.org

L. Herr
Pacific Interface Inc., Oakland, CA, USA
e-mail: laurin@pacific-interface.com

J. Schulze
University of California, San Diego, La Jolla, CA, USA
e-mail: jschulze@ucsd.edu

## 1 Introduction

There are two challenges in building distributed repositories. One is the establishment of a safe digital preservation environment in which data can be reliably stored and accessed over the long term. The second is the dynamic update of policies and their enforcement across collaborating institutions. How can policies in each system be enforced in a virtual organization (VO) when multiple software systems are deployed to manage massive data? Policy-driven systems such as the integrated Rule-Oriented System (iRODS) use a distributed rule engine to impose separate sets of policies at each storage location. The rule engine controls the procedures that are executed at each storage location, ensuring that data can only leave the storage system when all of the policies have been enforced. This approach is able to handle dynamic update requirements and enforce policies across collaborating institutions.

A policy-driven system builds a highly configurable environment with the ability to change the system behavior without a need to reimplement the software. Early in 1994, Morris Sloman presented an object-oriented model for building a policy-driven management system in which policies are viewed and constructed as policy objects [1,2]. With such an object-oriented approach, changing of policies is managed through creating, deleting, storing, and retrieving policy objects. Recent efforts in building policy-driven system include iRODS and the work by Alspaugh's team [3]. All of these approaches use a rule engine to support policy enforcement.

Aiming at providing a middleware solution for distributed data management, iRODS [4–7] was initially developed to replicate the capabilities of its predecessor, the Storage Resource Broker (SRB) [8–10]. iRODS addresses many important issues related to digital preservation through use

Published online: 27 January 2012

of a distributed storage model, including data replication, adaptive interfaces for new storage technologies, descriptive preservation metadata, and the capabilities of self-healing and disaster recovery for damaged data. One important technology introduced in iRODS is the micro-service model, which makes it possible to dynamically compose the procedures controlled by a policy. In contrast to the object-oriented approach, the enforcement of each policy in iRODS is accomplished by linking a chain of micro-services. New micro-services can be plugged into the system to support new data formats. The micro-services are executed at the remote storage location under the control of the remote rule engine. Information that is passed between micro-services is stored in structures in memory, improving the efficiency compared to web service models. The information structures can be serialized for transmission over a network to another storage location, making it possible to support distributed computing. In this way, the iRODS system provides an excellent middleware solution for building a policy-driven system to manage digital repositories. The iRODS software is open source software, distributed under a BSD license and can be downloaded from http://irods.diceresearch.org.

The advantages of using the iRODS' micro-service model to enforce institutional policies can be summarized as follows:

- Procedures can be composed from fundamental operations encoded in micro-services. The procedures invoked by a policy can be dynamically updated, making it possible to update policies without having to change any software systems.
- Policies can be enforced across low level system calls, including I/O calls in any storage system supported by iRODS (UNIX, HPSS, Mac, and Windows).
- Procedures can be invoked at the storage system, avoiding the need to move data over a network. For massive data collections, this is essential in optimizing administrative functions.

A standard approach to improve data integrity and protect against data loss is to rely on the use of a Redundant Array of Independent Disks (RAID) [11]. As all system administrators know, a RAID system may not protect against multiple simultaneous disk failures or accidental data purge. Basically, periodic data backup to tapes is the primary solution for prevention of data loss. However, even tape systems can lose data. As observed in large data centers in the TeraGrid [12], operator error is the dominant source of data loss through tape overwrite. To protect against operator error, a second copy is needed at an alternate site on an alternate storage technology. The iRODS policy for data loss prevention is to make a second copy for each file to be protected. Notice in large data centers or data grids that deal with petabytes of data, periodic data backups are not possible. Instead, replication on ingestion is used to improve data reliability.

Within the Policy-Driven Repository Interoperability (PoDRI) and CineGrid projects, the iRODS middleware is used to implement distributed repositories to handle digital preservation and disaster management. The principal goal of the PoDRI project is to investigate interoperability mechanisms between iRODS and Fedora repositories at the policy level. We seek an approach in which iRODS adds a digital preservation layer to the Fedora system [13,14] that enforces policies, including ingestion, administration, preservation, assessment, and access procedures. CineGrid is a global consortium of universities, public and private sector institutions, corporations, and high-speed network providers [15,16]. Under the National Digital Information Infrastructure and Preservation Program (NDIIP) funded by the Library of Congress, CineGrid has partnered with Academy of Motion Picture Arts and Sciences (AMPAS) to prototype a digital preservation solution to meet challenges in archiving digital material for the Academy Film Archive [17]. Built upon iRODS technology, distributed CineGrid repositories have been established at CineGrid partner sites at the University of California in San Diego, the Czech Republic National Research and Education Network, the Electronic Visualization Laboratory at the University of Illinois at Chicago, the Japan Keio University Research Institute for Digital Media and Content, and the University of Amsterdam, Netherland.

In this article, we will first discuss, in general, the requirements for establishing a generic digital preservation environment that can manage massive datasets for both the CineGrid and PoDRI projects. We will then give an introduction to iRODS and its micro-service model. We will demonstrate how policies can be turned into computer actionable rules that control the execution of procedures composed from micro-services. Three main policies for data ingestion, deletion, and automatic replication (so-called self-healing capability) will be discussed in detail. We will describe a policy execution model to demonstrate how an interoperability mechanism can be constructed to enforce iRODS policies for Fedora. And finally, we will present recent research results from the PoDRI project and describe a proof-of-concept model using two digital library technologies to jointly manage a digital repository, by combining two technologies through policy-level integration.

## 2 Digital preservation environment

As we discussed above, data loss may occur across all types of storage systems. This risk is certainly true for repositories within PoDRI and of CineGrid. The question is, with the existing hardware infrastructure including storage and network connections, if we can find a solution to build a

safe environment in which digital data can be preserved for the long term. It will be helpful to give a detailed discussion of the requirements for building durable and safe digital repositories and the solutions that iRODS can offer through a distributed data management approach along with its micro-service execution model.

### 2.1 Data model

iRODS provides two data models. One is the iRODS managed data content, in which all data files are stored inside iRODS vaults and can be only accessed through iRODS interface. Another one is the virtual data link model in which external files are registered into an iRODs data grid without physically making a copy. In some cases, the original data owners wish to keep the data files on their original disks. In this case, data files can be registered into iRODS as the prime copies. In this model, the files can be accessed both as a regular files in disks and through iRODS.

iRODS maintains a logical name space for each digital object as a persistent object identifier. Based on this naming method, an iRODS URI scheme has been implemented in the iRODS' Jargon Java toolkit that can be used for integration with other software system. In the PoDRI project, the iRODS URI scheme is used to virtually register iRODS objects, including digital content, system metadata, and user-defined metadata, into the Fedora system [18,19].

### 2.2 Data replication

Data replication is the key mechanism iRODS uses to handle digital preservation challenge in a distributed environment. To maintain the data integrity of a file, duplicated copies (at least three), or replicas, of each file must be kept in repositories in different locations on different storage systems. Note that the crucial file replication operation must be supported by a high throughput bandwidth network. And in the software layer, support for parallel file transfer is highly desired. To accelerate file transfer, iRODS has implemented both parallel file transfer using multiple TCP/IP streams and the Reliable Blast UDP protocol.

### 2.3 Adaptive interface for new storage technology

The preservation environment should be capable of integrating new types of storage into the system. iRODS has a flexible storage driver interface that maps from standard POSIX I/O operations to the protocol required by different types of storage systems such as UNIX, HPSS, and Windows. For a new storage system, a new driver can be developed and plugged into the iRODS framework. The iRODS storage interface supports most system-level storage APIs such as open, close, read, write, seek, stat, and sync in UNIX.

### 2.4 Metadata

Representation information is a type of preservation metadata that describes the provenance, structure, and procedures that can be applied to records. A preservation environment must be capable of managing multiple types of representation metadata [20,21]. The iRODS has a flexible metadata model that supports triplets of attribute name, attribute value, and attribute unit (optional), which are stored in the iCAT catalog. Each collection and even individual records may have a unique set of metadata. In addition, an XML data model has been incorporated within iRODS to support more complex metadata structures and migrate metadata between different representation standards. Note that all state information generated within iRODS is accessible as metadata on one of five logical name spaces for files, users, storage systems, rules, and micro-services.

When iRODS is integrated with other software systems such as Fedora, the ability to cross-register metadata between the systems is desirable. State information created within iRODS should be accessible from Fedora. Representation information generated in Fedora should be accessible from iRODS. As mentioned in the Sect. 2.1, both iRODS system metadata and user-defined metadata can be exposed to other software systems through a URI scheme, which can be used to register iRODS metadata as datastreams in Fedora objects.

### 2.5 Self-healing capability

When maintaining several copies for each digital object, a preservation environment should be capable of self-healing. The system should automatically detect corrupted copies and make necessary repairs. This requires an integrity mechanism such as checksums, synchronization mechanisms to manage updates to replicas, query mechanisms on the iCAT catalog to identify files within a collection, and iteration mechanisms to loop over the files. A micro-service has been developed to handle the complex logic in this self-healing function, which will be described in detail in Sect. 5.3.

### 2.6 Disaster management

Through iRODS, CineGrid is managing a collection that has been distributed and replicated across storage locations within the U.S., Netherlands, Czech Republic, and Japan. Through periodic validations of the integrity of the replicated collections, a true disaster-proof environment for CineGrid digital asset has been created.

### 2.7 Security

Currently, iRODS can use three authentication mechanisms, ranging from a challenge-response method, to Public Key

Infrastructure (PKI) authentication based on Grid Security Infrastructure [22], to Kerberos-based authentication.

The iRODs peer-to-peer servers are installed at the application level under a regular user account. The implication is that the data at that storage location are owned by the Unix user account. iRODS employs trust virtualization, to conduct operations on behalf of the user at the remote storage location. iRODS authenticates each user independently of the storage system, checks authorization for each operation, and then authenticates to the remote server to perform the operations on behalf of the user. This is a single sign-on environment, in which the user does not require a separate account at each location where data are stored. It is highly recommended that the local site administrator set read and write permission for the iRODS user account that runs the iRODS server to an explicit vault area only. In this way, the iRODS server is sandboxed in a restricted area(s).

When registering a virtual data link, the iRODS server checks whether the user issuing the iRODS file registration command is the owner of the file to be registered into iRODS.

### 2.8 Audit trail

In order to validate chain of custody, a preservation environment needs the ability to track all operations performed upon a record, including who issued the operation and when the operation was executed. When deploying iRODS servers, there is an option to turn on audit trails. The iRODS audit trail function records a wide range of information including all operations performed on data files and collections. For a complete list of the audit trail information managed within iRODS, please check out the iRODS web site at http://www.irods.org.

### 2.9 Reconfigurable policy implementation

Whether an object-oriented approach or micro-service-oriented approach is used, the goal is to build digital repositories in which policies are not hard-coded but can be deployed dynamically. Policy deployment is simplified when policies can be grouped into modules, when execution priority can be set between policies, and when more than one policy can be applied on a given operation. In the next two sections, we will introduce the micro-service model, and provide examples of mapping policies to computer actionable rules to demonstrate how this requirement is handled in iRODS.

## 3 About iRODS

Based on the SRB technology, iRODS is second generation middleware developed by the Data Intensive Environments (DICE) group at the Univ. of North Carolina, Chapel Hill and Univ. of California, San Diego. iRODS provides the data management functions needed in a distributed environment such as a file transfer service, data replication services, and metadata management services. iRODS supports scalable solutions for collections ranging from a few gigabytes- to petabytes-sized collections comprised from hundreds of millions of files. Applications range from data grids to share data, to digital libraries, to preservation environment, to data processing pipelines.

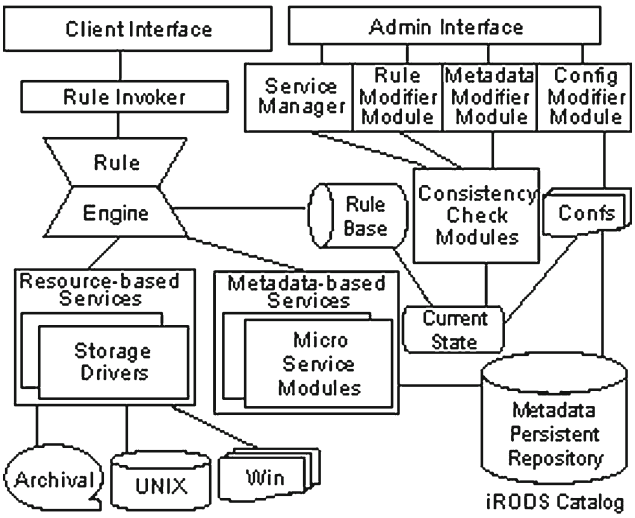


**Fig. 1** A Schematic diagram of the iRODS software architecture

As is shown in Fig. 1, iRODS drivers are the layer of software that access heterogeneous storage systems such as UNIX files and HPSS tape systems. Within this layer, iRODS uses a well-defined set of I/O calls that are POSIX compliant and can be easily used to adapt to any new storage hardware. A metadata catalog called iCAT, is managed within a relational database and is accessed through a special iCAT-enabled server. The iCAT stores information about datasets, the distributed storage systems, user accounts, etc. Files in iRODS are organized inside a hierarchy of logical collections that is similar to the UNIX directory system. The difference is that UNIX files reside on mounted disks while iRODS stores the files across distributed machines. A virtual organization managed by a catalog is called an iRODS zone. Different zones can talk to each other through the iRODS federation mechanism. Policies can be defined to control the sharing and access of files across zones once certain permissions are set properly. The capability of managing distributed storage provides an ideal solution for the CineGrid project in which movie or media files from different project collaborators are organized into a logical shared collection. The flexibility of the iRODS storage driver model allows different types of storages to be integrated, forming a sustainable digital preservation environment that can incorporate new storage technologies when they become available.

## 4 Micro-services

In iRODS, data management operations are encapsulated within micro-services that are implemented in C [23]. Each micro-service is intended to be a simple function that can be composed with other micro-services to implement a procedure. The micro-services can exchange information through parameters, through structures in memory, or through communication over a network to another iRODS server. The specification of an iRODS rule is defined as a chain of micro-services and rules. The rules are currently stored in an ASCII file. This allows system administrators to change the behavior of the data management system without the need to recompile the software. Various combinations of micro-services can transform an iRODS data grid from a data sharing environment into a digital library or a preservation environment. This malleability makes it possible to use a generic iRODS framework to implement a wide variety of data management applications.

Micro-services are provided that implement traditional workflow mechanisms such as queries on the metadata catalog, loops over query results, conditional execution, simple mathematic functions, and error recovery. The rules that control the execution of the micro-services are triggered when specific locations within the iRODS framework are executed. Sixty-four policy hooks are provided to control manipulation of files (creation, deletion, replication, migration), control creation and deletion of users, control creation and deletion of storage systems, etc. Hooks are provided to execute both pre- and post-processing policies. Pre-processing policies typically provide additional permission controls, extraction of metadata, and creation of derived data products, post-processing policies typically control redaction, delayed operations, and creation of presentation products.

### 4.1 iRODS micro-service model

A wide range of micro-services are available for creating procedures within iRODS. They include functions to manipulate data objects, collections, the iCAT catalog, metadata, XML processing, and storage resources [24]. There are also many specialized micro-services that are used to integrate libraries such as the Hierarchical Data Format (HDF) data sub-setting routines, to access remote web services, to execute XSLT transformations, and to manipulate containers such as tar files.

Micro-services are controlled by iRODS rules, each of which is composed of one or a chain of one or more micro-services. A rule engine inside iRODS is responsible for executing the workflow of chained micro-services [25,26]. The syntax for an iRODS rule is defined by four parts: the action name which identifies the hook within the iRODS framework where the rule will be invoked, a condition that must be met for the rule to execute, the chain of micro-services and rules to be executed, and a recovery chain of micro-services]. These four components are combined to form into a single string separated by "|."

actionDef | condition | workflow-chain | recovery-chain

In the case of deploying a rule that will be executed at one of the policy management hooks in the iRODS framework, the reserved action name should be used. For example, "*acPostProcForPut*" is the reserved name for post-processing rules applied after an object is uploaded into iRODS.

### 4.2 Micro-service versus workflow

The micro-service model is, in essence, a server-side workflow. Traditional scientific workflow systems such as Kepler [27], move the data to a compute server where the workflow actions are applied. A scientific workflow typically consists of modules that execute a computational algorithm, image processing, data parsing, etc. These are basically software applications (or business logic procedures in industry) that generate file-based output, or a simple set of output parameters. In the iRODS server-side workflow, however, sophisticated structures can be generated by a micro-service and stored in memory for use by the next micro-service in the chain. This makes execution of simple functions much more efficient within iRODS than in traditional workflows or web services. Default data structures are defined for each of the output structures used by the micro-services. In effect, the iRODS framework not only supports dynamic composition of workflows but also explicitly identifies and manages all data structures generated by each micro-service.

### 4.3 Automating administrative tasks

Rules can be designed to automate administrative tasks such as migration of data to new storage systems, execution of retention schedules, and monitoring of system status. Rules can be deferred for execution at a later time, such as a post-processing rule for creating a replica on a tape archive. Rules can also be executed periodically, such as verification of integrity, verification that the right number of copies are present, and verification that the correct file distribution is in effect.

## 5 Mapping policies to software implementation

When constructing distributed repositories, it is desirable for system administers to have the ability to reconfigure the system to automate administrative tasks and enforce local policies. In the CineGrid project, three policies for deletion

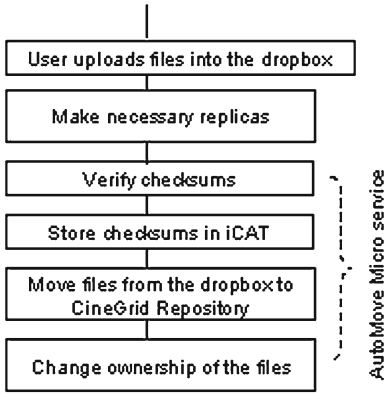


**Fig. 2** Data ingestion process

control, data ingestion control, and digital preservation control, were discussed and designed in detail. In this section, we will present three examples to demonstrate how to design iRODS rules using micro-services to enforce CineGrid policies. For the digital preservation policy, a new micro-service was developed and plugged into the iRODS release. This micro-service implements a self-healing function for digital objects within CineGrid. It can also be used by other data management systems that require a similar functionality. The iRODS environment enables reuse of workflow components, simplifying the creation of new data management applications. The iRODS environment also makes it possible to create generic rule sets that can be modified to include the specific policies of other data management applications. We also present a framework for modeling and executing iRODS rules from Fedora to demonstrate an approach that can be used by other data management systems for integrating with the iRODS policy-based data management.

### 5.1 Data ingestion policy

CineGrid repositories are distributed internationally. A data contributor can deposit his valuable media digital content into a designated repository once the content is approved by the CineGrid curator. CineGrid developed a data ingestion policy that addresses the followings:

- Each data contributor is given a designated place, called a dropbox, to upload their media files.
- The data integrity must be verified once a file is inside CineGrid.
- The system automatically moves a data file into its official place, change the ownership from original data contributor to CineGrid curator, and make the required replicas.

When using iRODS, the first two bullets can be taken care of by using the checksum verification option when uploading a data file into its designated dropbox. As is shown in Fig. 2, operations mentioned in the third bullet are carried out by a new micro-service, "*msiDataObjAutoMove*." The data ingestion rule is deployed through the reserved action, "*acPostProcForPut*," which is a trigger for post-processing after a data file is created (or uploaded) inside iRODS, and the micro-services, "*msiDataObjAutoMove*" and "*msiDataObjRep*:"

```
acPostProcForPut | $objPath like /Cine/home/user8/dropbox/* | msiDataObjRepl ($objPath, CineGrpResc,*st) ## msiDataObjRepl ($objPath, CineGrpResc, *st) ## msiDataObjAutoMove ($objPath, /Cine/home/user8/dropbox,/Cine/CinegridContent/CogScience, CineCurator, true) | nop#nop#nop
```

In this example, any file uploaded by the "*user8*" in his dropbox will be first replicated twice into a group resource, "CineGrpResc." It then calls the "*msiDataObjAutoMove*" micro-service to move the data file along with its replicas to the destination collection, "*/Cine/CinegridContent/CogScience*." The "*CineCurator*" is the user account used by CineGrid data curator and will be the new owner for the newly uploaded data file. The "true" value for the last parameter tells the micro-service to compute the (MD5) checksum and store it in the iCAT catalog if the system has not already done so. The "##" is used to chain micro-services in sequential order according to iRODS rule syntax [28]. This rule is activated once it is added to the "*core.irb*" file, the main rule configuration file for an iRODS server.

### 5.2 Deletion policy

Once a digital file has been approved and becomes an official asset inside a preservation system, the file should never be deleted. This means that a rigid policy of no deletion for any official media files must be enforced inside the iRODS data grid. When files are managed in a regular disk, there is still a possibility that they can be deleted accidentally by a privileged user using a forcing option. However, this risk can be avoided when all digital assets are stored and managed by iRODS middleware. In iRODS, there is a built-in action, "*acDataDeletePolicy*," which is a trigger for an iRODS rule to be executed before a file is deleted. This built-in action can be used to invoke an existing micro-service, "*msiDeleteDisallowed*," to form a rule that enforces the rigid deletion denial policy. The deployment of the rule is simply a text entry in "*core.irb*:"

```
acDataDeletePolicy | $objPath like /Cine/CinegridContent/*
| msiDeleteDisallowed | nop
```

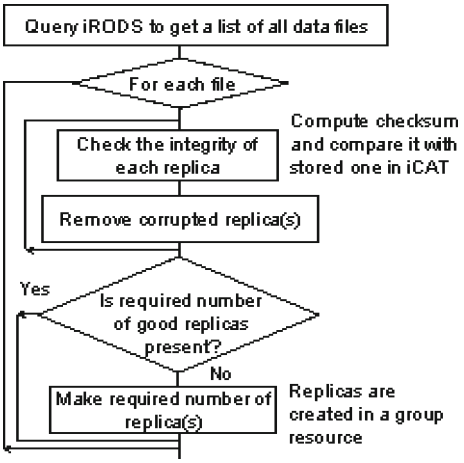


**Fig. 3** Digital preservation policy specifies a procedure for automatic repair for corrupted data files

In this example, a deletion denial policy has been implemented for all objects under the collection "*/Cine/CinegridContent*." The "*$objPath*" is a default logical name for the iRODS internal object path variable. The "*nop*" is a reserved word representing a null value in iRODS' rule language. In this example, it means no recovery chain is defined. Notice that the rule can be easily disabled by commenting out the line in "*core.irb*." The management of versions of the core.irb file is necessary to ensure that a consistent rule set is being run.

### 5.3 Digital preservation policy

The enforcement of integrity is an essential property of a preservation environment. As we mentioned in Sect. 2.5, it requires that the system have a self-healing capability, meaning that the system periodically checks the integrity of digital assets and makes necessary repairs when it finds corrupted copies.

To ensure the data integrity inside repositories, a certain number of replicas is maintained so that the system can make necessary repairs from a good copy. It is usually recommended to have at least three copies to minimize the risk of losing data. A digital preservation micro-service, called "*msiAutoReplicateService*," runs periodically inside iRODS to protect data loss due to possible corruption of data files. Fig. 3 shows the procedure of the micro-service. For each replica of a file, it computes the checksum and compares the result with the checksum stored in iCAT catalog. If a mismatch is found, it deletes the replica and creates a new replica from a good copy in another storage place.

Here is an example that checks the collection "*/Cine/CinegridContent*" weekly ensure that a minimum of three replicas have been created in the group resource "*CineGrpResc*."

```
DigitalPreserve || delayExec(<ET>6</ET><EF>7d</EF>,
msiAutoReplicateService(/Cine/CinegridContent, true, 3,
CineGrpResc, null), nop) | nop
```

In this example, "*DigitalPreserve*" is the name of the rule. There is no condition for the execution of this rule. The "delayExec" is a micro-service that is used to execute a delayed or a periodic rule. Note that the service can also be executed on a refined garrulity and in parallel by breaking a top collection into sub-collections and deploying the service multiple times (multiple entries in "*core.irb*") for the sub-collections.

### 5.4 Enforcing iRODS rules in Fedora

One of the goals of the PoDRI project is to investigate the interoperability of policies between Fedora and iRODS. When iRODS serves as a storage layer for Fedora, for example, it will be extremely useful if the above digital preservation rule can be configured and deployed from Fedora by defining the minimum number of copies and the time between tests.

There are four types of digital objects in Fedora: data objects, service definition objects, service deployment objects, and content model objects [29,30]. A service definition or SDef object defines a set of operations that can be performed for certain types of data objects. A service deployment or SDep object specifies where and how to execute the function. A content model (CModel) object performs dissemination for a type (or class) of data objects in Fedora.

The "iRule" is an iRODS UNIX client command for interactive execution of a rule by an iRODS server. A Web service can be developed with the iRODS Jargon toolkit to act as a client for iRODS servers to invoke a rule operation, i.e., performing the "iRule" function. A simplified schematic version of input message for the "iRule" web service can be one of the following:

```
<message name= "iruleRequest">
    <part name= "actionDef" type= "xs:string"/ >
    <part name= "condition" type= "xs:string"/ >
    <part name= "workflow-chain" type= "xs:string"/ >
    <part name= "recovery-chain" type= "xs:string"/ >
</message>
```

Three objects, "iRule-ServiceDef," "iRule-ServiceDep," and "iRule-ContentModel" for disseminating iRODS rules can be created based on this Web service. Each iRODS rule can be mapped into Fedora as a data object that has a relation for "hasModel" pointing to the content model, "iRule-ContentModel." With this association, the data object is considered as an iRule type object in Fedora. It also has a datastream that is an XML document containing iRODS rule parameters

described in "iruleRequest" message. The "iRule-Service-Def" has an operation, "iRule-Execute," that is the access point for the datastream.

The web service needs another input message to specify the storage policy, which generically targets at all storage systems including UNIX, iRODS, database, etc. This message specifies the target storage place to deploy the action of the policy. In the iRODS case, the input message contains basically iRODS storage configuration, including hostname, port number, iRODS storage resource, plus the user login information.

## 6 Technology fusion

Led by Professor Richard Marciano at the University of North Carolina (UNC), the PoDRI project has been investigating cross-repository policy-level interoperability, mainly between iRODS and Fedora. The project team consists of researchers from the Sustainable Archives & Leveraging Technologies (SALT) lab, the School of Information and Library Science, and the Libraries at UNC, iRODS team members from UNC and UCSD, and a senior researcher from the Fedora team. The research focus of the project aims at finding answers of the following [31]:

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

The high-level investigation of the two technologies presents a new horizon to build digital repositories. It allows us to extract the key requirements and structure of digital library technology, including digital object modeling, management of preservation information, and configurable policy enforcement. Pcolar et al. [31] conducted comprehensive research for both iRODS and Fedora and presented a set of "conceptual operations" for a wide range of policy-level integration of iRODS and Fedora, from data structure, storage module, management of metadata, audit trail, provenance information, and policies. Note that policies are expressed as Fedora Digital objects (FDO), stored and preserved in the corresponding repository, as described in Sect. 5.4.

An exciting prototype is deploying both iRODS and Fedora simultaneously to construct and manage a digital repository as depicted in Fig. 4. iRODS is mainly used as a rule engine for policy implementation and Fedora provides rich data modeling semantics such as complex objects and flexible metadata modeling.

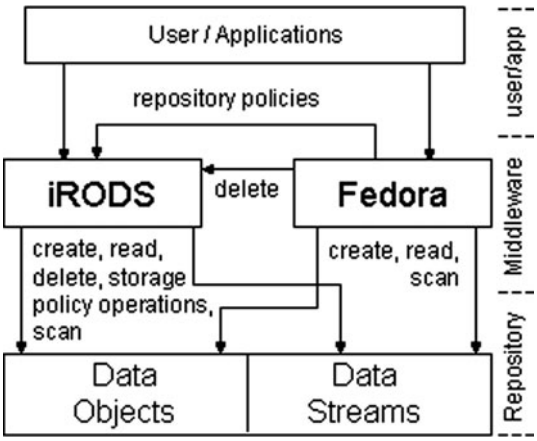


**Fig. 4** Managing a digital repository using a hybrid model of iRODS and Fedora

Compared with iRODS, Fedora provides a flexible and extensible object model through its FOXML schema. It can easily incorporate any descriptive digital preservation metadata standards such as PREMIS and METS. For the data structure in the repository to store digital objects, Pcolar et al. even described a way to model the iRODS hierarchical data structure through its content modeling method in Fedora. On the other hand, iRODS has a configurable micro-service model that is used to map policies to actionable computer code to enforce policies in both system level and user level as described in Sects. 4 and 5. iRODS provides a powerful policy engine in the proposed hybrid approach to enforce digital preservation policies such as data replication and data integrity check. Another advantage of iRODS is its capability of handling large digital objects such as movie files in a distributed environment. The two systems can not only co-exist to jointly manage a digital repository but also complement each other's functions.

With Fedora's method of storing datasets, a digital repository consists of two main collections of FDOs and datastreams. An important concept in Fedora for the relationship between digital objects and their catalog is that the catalog can be re-built anytime by a Fedora's walk-through function. The function scans an entire collection of digital objects and reconstructs the catalog. In the proposed hybrid model, a digital object, or a dataset, can be created through either iRODS or Fedora. When a dataset is created through iRODS, the periodic Fedora walk-through function will automatically pick up the new dataset and register it in Fedora's catalog. Similarly, there is a need to develop a similar walk-through function in iRODS to register any digital objects created through Fedora.

In the proposed hybrid model, each system can independently manage a repository. This means that the repository does not depend on each individual technology. In other words, when one of the technologies is obsolete or depre-

cated, the repository can be continuously managed by another technology and thus, in some sense, the repository can be decoupled from a particular technology.

## 7 Summary

In constructing distributed digital repositories, a policy-driven system provides an ideal solution to update and enforce preservation policies. The iRODS middleware provides the essential functions that are needed to manage distributed data. Through the iRODS micro-service oriented model, policies for managing distributed repositories can be dynamically reconfigured, constructed, and enforced. Easy system integration of iRODS with other software systems allows policies to be designed and enforced through a web service interface. The research development in the area of the policy interoperability between iRODS and Fedora demonstrate that both technologies can be combined to construct and manage a digital repository that provides flexible object modeling and a configurable policy oriented system in constructing a digital preservation environment.

**Acknowledgments** This study was supported by a Research and Demonstration grant from the Institute of Museum and Library Services (IMLS LG-06-09-0184-09), "Policy-Driven Repository Interoperability," a National Science Foundation grant, "Software Development for Cyber Infrastructure," and CineGrid Exchange Development project.

## References

1. Sloman, M.: Policy driven management for distributed systems. J. Netw. Syst. Manag. **2**(4), 333–360 (1994)
2. Moffett, J.D., Sloman, M.S.: The representation of policies as system object. In: Proceedings of Conference on Organizational Computer Systems, Atlanta, SIGOIS Bulletin, vol. 12, pp. 171–184 (1991)
3. Alspaugh, S., Chervenak, A., Deelman, E.: Policy-Driven Data Management for Distributed Scientific Collaborations Usings Rule Engine, SC08. Austin, Texas. 15–21 November 2008
4. Moore, R., Rajasekar, A., Wan, M., Schroeder, W.: Policy-based distributed data management systems. In: The 4th International Conference on Open Repositories, Atlanta, Georgia, 19 May 2009
5. Introduction to iRODS. https://www.irods.org/index.php/Introduction_to_iRODS. Accessed 2 Feb 2010
6. Hedges, M., Hasan, A., Blanke, T.: Management and preservation of research data with iRODS. In: Proceedings of the ACM first workshop on CyberInfrastructure: information management in eScience, pp. 17–22 (2007)
7. Moore, Reagan W.: Towards a Theory of Digital Preservation, IJDC, vol. 3, June 2008
8. Storage Resource Broker. http://www.sdsc.edu/srb/index.php/Main_Page. Accessed 2 Feb 2010
9. Rajasekar, A., Wan, M., Moore, R.: MySRB & SRB—Components of a Data Grid. In: The 11th International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, 24–26 July 2002
10. Moore, R., Jagatheesan, A., Rajasekar, A., Wan, M., Schroeder, W.: Data grid management systems. In: Proceedings of the 21st IEEE/NASA Conference on Mass Storage Systems and Technologies (MSST), College Park, Maryland, 13–16 April 2004
11. http://www.datarecovery.com.sg/data_recovery/types_of_raid_configurations.htm. Accessed 2 Feb 2010
12. http://www.teragrid.org. Accessed 2 Feb 2010
13. Fedora Commons. http://www.fedora-commons.org. Accessed 2 Feb 2010
14. http://en.wikipedia.org/wiki/Fedora_Commons. Accessed 2 Feb 2010
15. http://www.cinegrid.org. Accessed 2 Feb 2010
16. Laurin Herr. CineGrid@GLIF 2009. http://www.glif.is/meetings/2009/rap/herr-cinegrid.pdf. Accessed 2 Feb 2010
17. The Digital Dilemma. http://www.oscars.org/science-technology/council/projects/digitaldilemma/index.html. Accessed 2 Feb 2010
18. Zhu, B., Marciano, R., Moore, R.: Enabling Inter-repository Access Management between iRODS and Fedora. In: The 4th International Conference on Open Repositories. Atlanta, Georgia, 19 May 2009
19. https://www.irods.org/index.php/Fedora. Accessed 2 Feb 2010
20. Dappert, A., Caplan P., Guenther, R.: Digital Preservation Metadata. http://www.planets-project.eu/docs/presentations/Dappert_PreservationMetadata.pdf. Accessed 2 Feb 2010
21. PREMIS Data Dictionary for Preservation Metadata. http://www.loc.gov/standards/premis/v2/premis-2-0.pdf. Accessed 2 Feb 2010
22. Grid Security Infrastructure (in iRODS). https://www.irods.org/index.php/GSI. Accessed 2 Feb 2010
23. Micro-Services. https://www.irods.org/index.php/Micro-Services. Accessed 2 Feb 2010
24. Released Micro Services. https://www.irods.org/index.php/Released_Micro_Services. Accessed 2 Feb 2010
25. Rules. https://www.irods.org/index.php/Rules. Accessed 2 Feb 2010
26. iRODS Rules. https://www.irods.org/index.php/Rules. Accessed 2 Feb 2010
27. The Kepler Project. https://kepler-project.org. Accessed 2 Feb 2010
28. https://www.irods.org/index.php/Rules. Accessed 2 Feb 2010
29. http://fedora-commons.org/confluence/display/FCR30/Fedora+Digital+Object+Model. Accessed 2 Feb 2010
30. http://www.fedora-commons.org/documentation/3.0/userdocs/digitalobjects/objectModel.html. Accessed 2 Feb 2010
31. Pcolar, D., Davis, D., Zhu, B., Chassanoff, A., Hou, C.Y., Marciano, R.: Policy-Driven repository interoperability: enabling integration patterns for iRODS and Fedora. In: 7th International Conference on Preservation of Digital Objects (iPRES2010), Vienna, Austria, 19–24 September 2010