DataNet Federation Consortium Preservation Policy ToolKit

Reagan Moore
University of North Carolina at Chapel Hill
312 Lenoir Dr
Chapel Hill, NC 27599
919 962 9548
rwmoore@renci.org

Arcot Rajasekar
University of North Carolina at Chapel Hill
312 Lenoir Dr
Chapel Hill, NC 27599
919 966 3611
sekar@renci.org

Hao Xu
University of North Carolina at Chapel Hill
312 Lenoir Dr
Chapel Hill, NC 27599
919 962 9548
xuh@cs.unc.edu

ABSTRACT
The DataNet Federation Consortium uses a policy-based data management system to apply and enforce preservation requirements. This paper describes the Preservation Policy Toolkit developed by the consortium. In particular, the paper describes the infrastructure needed for preservation, presents examples of computer actionable forms of policies, and provides a generic template for designing actionable preservation policies.

General Terms
Preservation strategies and workflows.

Keywords
Policy-based data management, preservation policies, computer actionable procedures.

1. INTRODUCTION
The NSF DataNet Federation Consortium (DFC) infrastructure enables multiple Science and Engineering communities to implement their preferred data management applications and establish trusted research collaborations [1]. Partners within the DFC have implemented a variety of data-centric environments including data preservation systems (archives), data sharing systems, data publication systems (digital libraries), data distribution systems, and data processing systems (processing pipelines) to serve the needs of their specific communities and research groups. The DFC accommodates each type of data management application by specifying a set of policies that enforce the desired properties for that type of data management application:

- A trusted digital archive focuses on properties related to: authenticity; integrity; access control; chain of custody; persistent storage; fidelity; and original arrangement.
- A data sharing environment focuses on properties related to: unified name spaces for users, files, and collections; metadata-based discovery; access controls; auditing; hierarchical arrangement; and ease of access.
- A digital library focuses on: controlled name spaces for files, collections and metadata; descriptive metadata standards; standard data formats; multi-faceted search; and logical collection arrangements.
- A data distribution system focuses on: fault tolerance; automatic failover; on-demand caching; replication; synchronization; staleness control; high availability; streaming; and high-speed content delivery.
- A processing pipeline focuses on: controlled name spaces for users, files, collections, and procedures; distributed service and workflow automation; cloud computing; scheduling of high-performance computation; third-party and licensed service invocation; workflow reuse; repurposing of workflows; and provenance of workflows.

Each of these types of data management applications can build upon generic data grid infrastructure by choosing an appropriate set of policies and procedures. The DFC uses the integrated Rule Oriented Data System (iRODS) data grid software [2] as a platform to implement community-specific management policies. The policies determine when and where procedures are executed. Policies can be automatically enforced at policy enforcement points that are encoded in the software middleware within the iRODS system, or policies can be executed interactively by a user or grid administrator, or policies can be scheduled for deferred and periodic execution. The policy enforcement points typically control management policies. Deferred and periodic execution is used for administrative tasks. Interactive execution is used by users to launch remote workflows and is also used to validate assessment criteria.

The DFC is developing toolkits for each of the data management applications outlined above. This paper describes the Preservation Policy Toolkit (PPTK). The PPTK can be tuned, modified or extended by each Science and Engineering community to meet their particular requirements. In the next section, we describe the concepts behind the implementation of policies within iRODS, followed by a discussion of policy templates and policy languages, and summarize the elements in the PPTK. Several examples of policies are provided as part of the discussion.

2. POLICY CONCEPTS IN DFC
In this paper, we discuss the preservation environment needed to implement data management applications such as a trusted digital archive that automates policy enforcement within cyber-infrastructure. A preservation environment can be defined by the set of policies and procedures that enforce the properties of authenticity, integrity, access control, chain of custody, persistent storage, fidelity, and original arrangement. In Figure 1, a

generalization of this approach for implementing preservation properties is shown. Given a specific preservation purpose, an archive can be assembled that has desired properties such as integrity enforcement, arrangement, and access controls. The properties themselves may have associated requirements such as completeness (all files in the archive have the same property), correctness (incorrect values for information properties have been identified and isolated or eliminated), consensus (the properties represent the combined desire of the group assembling the archive), and consistency maintenance (the same metadata and data format standards have been applied to all files in the archive). Each desired property is enforced by a set of policies that determine when and where associated procedures are executed.

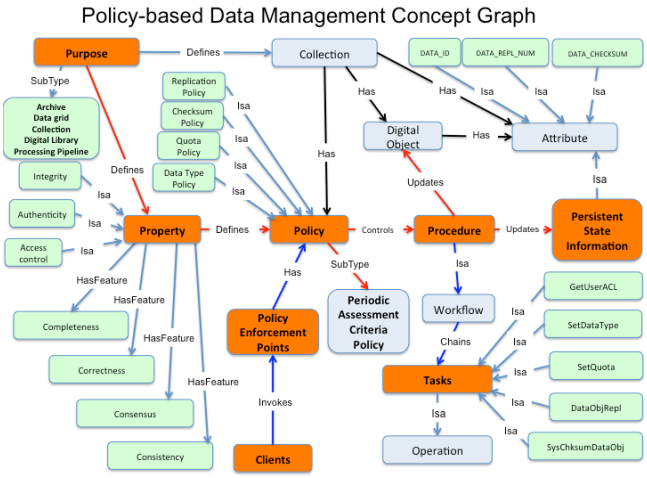

Figure 1. Policy Concepts

The associated procedures are implemented as workflows constructed by chaining basic tasks or functions (called micro-services) that are provided in the iRODS data grid. The functions implement basic operations such as generate a checksum, or replicate a file, or set the data type. The results of applying the functions are saved as persistent state information or metadata attributes on the name spaces for files, collections, users, storage systems, metadata, policies, and micro-services.

Consider the integrity maintenance property. In an implementation, one may perform such integrity maintenance by applying policies for generation of checksums and replication of files. A checksum is used as a digital signature to verify the fidelity and integrity of the deposited material in the archive. In some rigorous applications, more than one type of digital signature (using different algorithms) may need to be maintained as part of the digital collection. Replication is enforced to recover from disasters and failures. Periodic verification of checksums together with management of replicas provides a means to identify file corruption and rectify through synchronization with a high fidelity copy. Policies are needed to set the number of required replicas, set the verification periodicity, and define the mode of failure recovery. Additional policies apply this state information to enforce the integrity property when files are ingested into the archive.

In essence, policy-based preservation systems encapsulate four foundational concepts:

1. Purpose for creating the preservation archive expressed as the management of a set of desired properties.
2. Consensus on preservation enforcement as a set of desired policies.
3. Maintenance of preservation properties through a set of required procedures.
4. Tracking of preservation state information through required attributes assigned to the controlled name spaces for users, files, collections, and micro-services.

3. POLICY TEMPLATES

This view of preservation as the set of properties that will be maintained over time is consistent with the ISO 16363 standard [3]. Each of the trustworthiness metrics expressed by the standard can be captured in policies that are automatically enforced by the data management environment. The PTAB ISO 16363 Metric Knowledge Base lists a set of required supporting evidence for each metric. For example:

> "**4.6.1 The repository shall comply with Access Policies.**
> 1. Access policy for repository.
> 2. Collection Development Policy.
> 3. Definition of the Designated Community.
> 4. Demonstrations and discussion with relevant staff of what occurs when a query results in 'Access Denied'.
> 5. Documentation that illustrates the Access Policy is being carried out: Sign in sheets, logs of access, logs of successful and unsuccessful access to the system, follow up emails or help desk reports when 'access denials' received.
> 6. Examples of Preservation Description Information (PDI) that contain Access Rights information.
> 7. If there are access controls on private or restricted content, then particular events when the content was accessed by users or staff should be checked.
> 8. License agreements for content.
> 9. Mission Statement.
> 10. Relevant Copyright law.
> 11. Submission agreement(s).
> 12. User surveys or interviews that determine user satisfaction with delivery of DIP's."

Policies can be implemented through procedures that generate the required evidence for each metric. Policies can be created that identify the location of the required documentation, generate event information for each action, and log the results of all access checks. These policies can be implemented as machine-actionable procedures, enabling automation of preservation tasks.

A template that captures the information associated with a policy has been published by the Research Data Alliance Practical Policy working group [4]. In Table 1, an example is provided for specifying policies for access controls.

The example has two sections: the set of state attributes needed to decide when to execute the policy, and the set of state attributes needed while executing the policy.

The template can be used to design the set of controlling policies and execution procedures that implement the evidence specified for each ISO 16363 metric. The template lists the policy name, the constraints that limit application of the policy, the state information needed to evaluate the constraints, the operations that the policy will apply, and the persistent state information that is needed or changed by the policy.

The constraints imposed on the policy define how the policy should be applied. In this case, the archive may choose to enforce access controls by the role assigned to each user (administrator,

user) or by a unique identifier for each user (account name). The access controls may be applied at the collection level or at the individual file level. Choosing the type of access control to implement defines the state information that will be needed.

The operations performed for controlling access include:

- Creating identifiers for persons, collections, and files.
- Assigning roles to persons.
- Assigning access controls to collections and files (in effect a relationship between the person identifier and the file identifier).
- Assigning inheritance of access controls on collections (files can inherit the access control of the collection).
- Checking access permissions on reads and for other actions on the file.
- Verifying the set of access controls applied to files in a collection.

Table 1. Policy Template for Access Control

Policy type	**Constraint**	**State attributes for Constraint**
Access data	By role (type of person)	User_ID
		Role_type per User_ID
		Role_ACL
	By ACL (read permission)	User_ID
		File_name
		ACL per File_name per User_ID

	Operations	**State Attributes for Operation**
	Set person name	User_ID
		User_name
	Set file name	File_ID
		File_name
	Set role per person	User_ID
		Role_type
	Set ACL on file	File_ID
		User_ID
		ACL_type
	Set sticky bit on collection	Collection_name
		Sticky-bit_value
	Set access on replication	File_ID
		Replica_number
		User_ID
		ACL_type
	Execution - check ACL on read	File_name
		User_ID
		ACL_type
	Verify ACLs	File_ID
		Replica_number
		User_ID
		ACL_type

One can immediately notice that the evidence listed for the access control metric in ISO 16363 needs to be augmented with policies that are driven by the type of implementation. The preservation environment has to map from the metric evidence specification to the technologies that are currently available for implementation of the archive. Depending upon the choice of technology, different mappings will be required. For example:

- Choice of person identifier depends upon the type of authentication system that is used (certificate authority, LDAP directory, one-time password, ORCID).
- Choice of file identifier depends upon the type of storage system (Unix file system, tape archive, object store) and the object identifier (GUID, OID, handle, logical name).
- Choice of role-based or account-based access controls depends on the type of user authentication environment.
- Choice for identification of copies of files (replicas, backups, versions) depends upon the required persistence properties.

A second observation is that the documents specified in the audit checklist can be supported by generic policies. Thus policies for storing, finding, and retrieving documents can be used to archive the collection development policy, the definition of the designated community, examples of preservation description information (PDI) that contain access rights information, license agreements for content, mission statement, relevant copyright law, submission agreement(s), and user surveys or interviews that determine user satisfaction with delivery of DIP's. The document attributes may need to be organized and associated with either a user name space, or a collection name space, or individual files.

A third observation is that sign-in sheets, logs of access, logs of successful and unsuccessful access to the system, and follow up emails or help desk reports when 'access denials' are received can be supported by generic event management policies. If the archive is able to encapsulate information about all actions that are performed in standard events, then the events can be saved and indexed. A generalization of this is the ability to map from:

- An action that was taken (record ingestion, user access, archive administrator process),
- To the operation that was performed within the archive,
- To the state information change that resulted from the action.

It should then be possible to identify all interactions with the archive and verify that the resulting operations were consistently applied. This includes application of access controls, or maintenance of file integrity, or creation of AIPS, or tracking of submissions. An audit trail can be saved as the sequence of events that changed the archive state information. The events can be indexed and analyzed for compliance with the desired archive properties. In addition, all changes to the preservation environment state information can be correlated with a controlling policy.

In summary, multiple types of policies may be needed for each type of evidence:

1. Policies to set input parameters (environmental variables) needed for policy execution.
2. Policies to control execution of a procedure.
3. Policies to automate execution of administrative functions, typically performed by the archive administrator.
4. Policies to verify compliance with the desired preservation properties.

The policies may be run interactively by the archive administrator (policy type 1), or enforced at a policy enforcement point within

the software (policy type 2), or executed periodically by the rule engine (policy types 3 & 4). The policies are organized into a preservation policy toolkit. Each community that requires preservation can modify the policy toolkit to implement their required preservation policies.

4. POLICY VIRTUALIZATION

The choices made today for implementing an archive will change as better technology emerges. This raises an immediate challenge for preservation environments. How can the same policies be effectively applied in the future? How can the effort to migrate to new technologies be minimized? How can federation across multiple archive implementations be achieved? Note that migration to new technology and federation across heterogeneous technologies are effectively the same capability. At the point in time when new technology is acquired, both the old technology and the new technology will be present in the system. Records can be migrated from the old technology to the new technology using federation mechanisms. The ability to federate across technology implementations is essential for continued enforcement of preservation policies over time.

Policy-based data management systems such as data grids handle technology evolution through use of virtualization mechanisms. Interactions with technology are done through software middleware that map from the desired action to the protocol required by the technology choice. The software that does the mapping is encapsulated in a pluggable driver, enabling the replacement of the old technology by plugging in a driver for the new technology. Pluggable drivers are used within the DFC for interactions with authentication systems, storage systems, databases, network transport, rule engines, and micro-services (basic operations). Through plugins, a preservation environment can interact with multiple types of systems simultaneously, and manage migration to new technologies.

Virtualization also implements the ability to manage all of the properties of a preservation environment independently of the choice of technology. This includes management of the names of the users, the names of the files, the organization of files into collections, the provenance and descriptive metadata, the access controls, and administrative metadata such as checksums, file size and storage location. The information is stored as metadata in a database.

For example, consider the addition of a file to the system. Even though the explicit event is a simple file addition, the response of the system may require the execution of multiple policies, with each policy potentially executing procedures that manipulate multiple types of objects. Policies that are executed may include:

- Authentication of the person adding the file.
- Authorization for the addition of a file.
- Evaluation of a storage quota for the storage resource.
- Creation of a persistent identifier for the file.
- Validation of the Submission Information Package against the submission agreement.
- Logical arrangement of the file as a member of a collection (creation of a logical file name).
- Selection of a storage resource for the physical copy of the file.
- Creation of a physical file name on the storage resource
- Inheritance of access controls from the collection access controls.
- Creation of a checksum.
- Creation of a persistent object (storage of the file as received).
- Replication of the persistent object to a second storage location.
- Assignment of a retention period for the file.
- Assignment of a disposition procedure to the file.
- Assignment of a data type to the file based on the file extension.
- Creation of a copy with a required data format.
- Storage of system level metadata (owner name, access controls, checksum, file size, replica location, retention period, file type).
- Extraction and storage of descriptive metadata.
- Creation of an Archival Information Package (aggregation of metadata with the file into a container).
- Storage of the AIP.
- Replication of the AIP.
- Generation of event information for each step of the ingestion.
- Storage and indexing of the event information.

Policies can be defined that control each of the ingestion steps. It is then possible to associate different ingestion steps with different collections. Also the policies may need to evolve over time to handle changes in technologies, or changes in management, or changes in preservation standards. This will require support for multiple versions of policies, with different sets of constraints applied within each version. The policies will need to be archived along with the records to enable a future archivist to track how each record was controlled over time. It should be possible for a future archivist to start with an original Submission Information Package, apply the sequence of policies recorded in event information, and re-generate the current Archival Information Package.

4.1 State Information

Virtualization depends upon having a "complete" set of state information (metadata attributes) that can be queried and retrieved. Information is needed about the preservation environment for each step in the file ingestion process. This includes information about not only each record (representation information, provenance information, description information), but also information about the preservation environment (user names, storage locations, policies).

Typical file system state information is listed in Table 2. The information stored about each file is quite limited. A preservation environment augments this information with provenance information, representation information, description information,

Table 2. File System State Information

File Name
File Location on disk
Creation time
Modification time
File size
Access control
Locks
Soft Link
Directory

and administration information. In practice, the DFC manages more than 330 state information attributes about both the records and the preservation environment. Information is managed about users, files, collections, storage resources, metadata, rules, micro-services, quotas, system load, audit trails, and federations.

4.2 Operations

Virtualization depends upon having a complete description of all the operations that will be performed within the preservation environment. The operations performed upon a file system typically consist of create, open, close, read, write, update, seek, stat, chown, link, and unlink. Some of the operations may be applied to a file or to a group of files. Preservation environments require support for additional operations such as creation of checksums, replication, migration, and format transformation.

A generic characterization of operations performed within data management systems is needed. To base the discussion on well-known concepts, consider the characterization of file systems shown in Figure 2. The file system comprises an environment

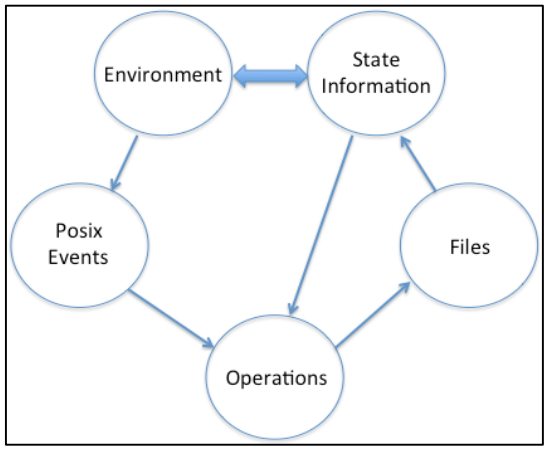

Figure 2. File System Characterization

that is defined by the state information maintained about each file. Interactions with the file system consist of events that specify an operation. Each operation manipulates a file and changes the associated state information. Operations may require access to state information such as file location, or file size, or file owner.

Interactions with the files are done through interactive execution of clients, which invoke the desired operation through a system call. This approach makes it possible to implement a standard data management interface on different types of hardware systems, which in turn enables the migration of files across storage systems.

We can generalize this model of data management by introducing policies that control the operations performed within the system. In Figure 3, we introduce three significant changes:

1) Operations are replaced by policies.
2) Files are replaced by objects.
3) Updates on objects and on state information are implemented as procedures.

Additional operations can be added to the system through the creation of new procedures. The knowledge needed to manage the procedures can be captured in policies, and the information needed to execute the new procedures can be added as additional metadata. This makes it possible to add operations to the preservation environment, along with the new policies and state information. The preservation environment can now evolve to track changes in preservation requirements, changes in technology, and changes in administration.

Within the DFC, procedures are implemented as workflows that are created by composing together basic functions, called micro-services. The DFC supports more than 300 micro-services that

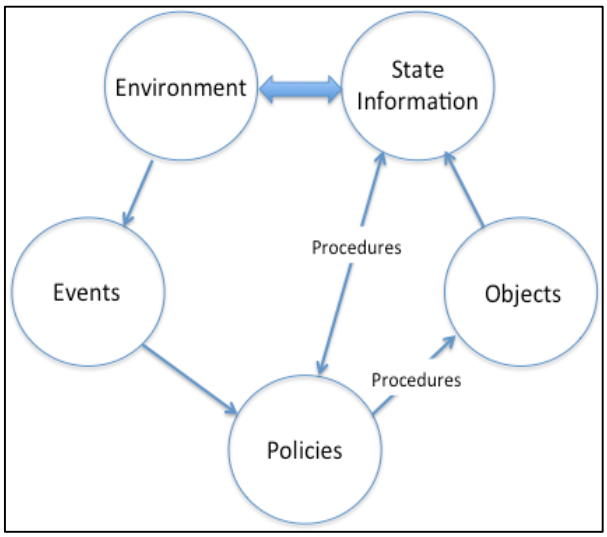

Figure 3. Policy-based Data Management

implement data management operations. The micro-services can be categorized as operations for user management, file manipulation, collection management, metadata manipulation, policy management, network management, messaging, administration (setting environment variables, quotas, load monitoring), and data grid manipulation (federation).

Micro-services can be created that support interaction with specific types of technology. A typical example is the creation of a micro-service that supports access to a remote service for file conversion. The micro-service manages the interaction with the network protocol required for communication with the remote service. Since multiple types of technology exist today, this requires support for versions of micro-services, as well as versions of state information.

5. POLICY LANGUAGE

Policies within the DFC are implemented as workflows, by chaining together micro-services. A rule engine is used to parse each workflow, evaluate the policy constraints, invoke execution of each micro-service, and manage errors. The workflow language had to be Turing complete, enabling the creation of workflows that included conditional tests and loop constructs. A typical policy would specify a constraint as a conditional test on system state information and session variables, generate a query that is sent through a catalog interface to a database, loop over the database query results, apply arithmetic operations and string manipulation to variables, and write results to standard out for interactive execution or to a file for storage within the data grid. In the DFC, new policies can be added dynamically to the system through inclusion in a rule base.

The choice of where and when to apply the policies is mediated through the use of policy enforcement points within the data grid software middleware. In the DFC, the locations of the original policy enforcement points were hard-coded. Through extensions developed by the iRODS Consortium [5], policy enforcement

points were made pluggable. Each time a new operation is added, pre-process and post-process policy enforcement points are added automatically.

A consequence of the micro-service plugin extension is that now every operation performed within the preservation environment can be tracked, along with the corresponding change to state information. The state changes can be saved as events that are indexed in an external indexing system. The events can be analyzed to verify compliance over time with the desired properties of the preservation environment.

The design of preservation policies that will be executable in the future is based on the assumption that the knowledge needed to interact with technology can be encapsulated in versions of micro-services. By invoking the current micro-service version, a policy will remain executable. This in turn requires that the preservation environment manage all information needed to apply the procedure within a metadata catalog, independently of the choice of storage technology. The catalog can then be queried to retrieve the information needed to apply the current micro-service version.

The policy language is interpreted by the rule engine. To enable long-term preservation, the rule engine itself had to be pluggable, enabling the use of a new rule engine and a new rule language by future archivists. The DFC preservation environment thus provides multiple levels of virtualization:

- From actions requested by clients to standard operations supported by the data grid.
- From the state information maintained by the data grid to the information required by the selected storage technology.
- From the knowledge encapsulated in micro-services to the execution of the standard data grid operations.
- From the standard operations supported by the data grid to the operations provided by the selected storage technology.
- From a consensus on management decisions to choice of policies enforced at policy enforcement points within the data grid.

With these levels of virtualization, a preservation environment can be created that is technology independent, enabling the incorporation of new technologies over time while maintaining persistent objects.

An example of the rule language is shown in Figure 5. Each workflow operation (variable assignment, string concatenation, foreach loop, conditional if test) is treated as a micro-service. The rule engine parses each line in the workflow, invokes the associated micro-service, and manages information exchanges between micro-services through in-memory data structures. The workflows can be distributed across multiple servers. Information exchange between servers is mediated by packing instructions that serialize the in-memory data structures, send the result over the network to the next participating server, and unpack the information into a local in-memory data structure in the remote system.

Policies are stored at each server in a distributed rule base. This improves performance, makes it possible to distribute the policy enforcement across all participating storage resources, and makes it possible to install different policy sets at each server. One consequence is that a distributed debugger is needed to analyze problems in distributed workflows. This capability is provided within the DFC infrastructure through use of a messaging system.

6. PRESERVATION POLICY TOOLKIT

The DFC has developed a set of policies required for preservation. The policies (forming a toolkit) are driven by community requirements and represent instances of computer actionable rules that control administrative operations. The policies are driven by local security requirements, local storage facilities, local authentication requirements, and local networking infrastructure. The examples provided in this paper are intended to illustrate some of the challenges in writing computer actionable rules. The rules are modifiable for application in other preservation environments.

6.1 Sample Policy: Network Firewall

Implementing policies for a preservation environment is a complex task. A standard challenge in implementing a preservation environment is management of network firewalls. If an archive storage resource is located behind a firewall, prohibiting access from external networks, then policies are needed to manage ingestion. One approach is to implement data staging, with records deposited into a network accessible storage system as shown in Figure 4.

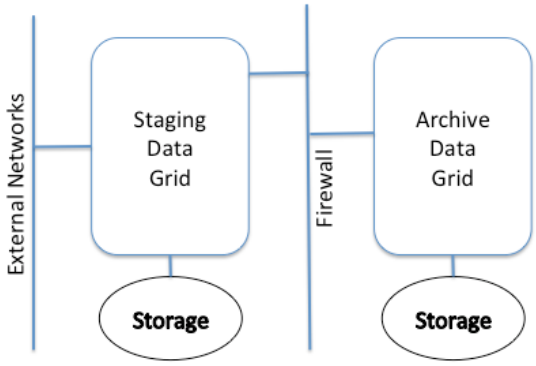

Figure 4. Deep Archive

A policy running within the Staging Data Grid analyzes the Submission Information Packages for compliance with a submission agreement, checks for the presence of viruses, and sets an approval flag for qualified data. A policy that runs within the Archive Data Grid queries the external Staging Data Grid and pulls the approved files into the archive.

A version of the staging policy that implements multiple operational steps needed for a production environment is shown in Figure 5. Files are copied from the staging area into an archive by a policy running on the staging area. The rule implements the following steps:

- Use session variables to find the data grid and account name under which files will be accessed.
  $rodsZoneClient is the name of the staging data grid.
  $userNameClient is the account name on the staging data grid.
- Create path names for the staging directory and the archive directory.
- Get the current system time in a human readable format.
- Check whether a directory exists in the archive for storing log files.
- Create the directory if needed.
- If the log directory cannot be created, fail with an error message.
- Create the log file for tracking data storage operations.
- Create a query to list the files and their checksums in the staging data grid.
- Execute the query and loop over the result set.

- Extract each file name and checksum value.
- Copy each file to the archive and force an overwrite of existing files.
- Set ownership access controls on each file.
- Calculate the checksum of each file after it is moved.
- Verify the checksum is correct.
- For files moved successfully, delete the copy in the staging area.

Variants of the rule are used to execute the rule from the archive and pull data from the staging area, initiate the original archive, and push data to a second archive.

```
myStagingRule {
# Loop over files in a staging area,
#/$rodsZoneClient/home/$userNameClient/*stage
# Put all files into collection
#/*DestZone/home/$userNameClient#$rodsZoneClient/*Coll

  *Src = "/$rodsZoneClient/home/$userNameClient/*Stage";
  *Dest= "/*DestZone/home/$userNameClient"
++"#$rodsZoneClient/" ++ *Coll;

#=get current time, Timestamp is YYY-MM-DD.hh:mm:ss  =====
  msiGetSystemTime(*TimeH, "human");

#=create a collection for log files if it does not exist ==========
  *LPath = "*Dest/log";
  *Query0 = select count(COLL_ID) where COLL_NAME = '*LPath';
  foreach(*Row0 in *Query0) {*Result = *Row0.COLL_ID;}
  if(*Result == "0" ) {
    msiCollCreate(*LPath, "0", *Status);
    if(*Status < 0) {
      writeLine("serverlog", "Could not create log collection");
      fail;
    }  # end of check on status
  }  # end of log collection creation

#= create file into which results will be written ================
  *Lfile = "*LPath/Check-*TimeH";
  *Dfile = "destRescName=*Res++++forceFlag=";
  msiDataObjCreate(*Lfile, *Dfile, *L_FD);
#========== find files to stage ======================
  *Query = select DATA_NAME, DATA_CHECKSUM where
COLL_NAME = '*Src';
  foreach(*Row in *Query) {
    *File = *Row.DATA_NAME;
    *Check = *Row.DATA_CHECKSUM;
    *Src1 = *Src ++ "/" ++ *File;
    *Dest1 = *Dest ++ "/" ++ *File;
# ==========Move file and set access permission =========

msiDataObjCopy(*Src1,*Dest1,"destRescName=*Res++++forceFlag=
", *Status);
    msiSetACL("default", "own", $userNameClient, *Dest1);
    writeLine("*Lfile", "Moved file *Src1 to *Dest1");
# ========== verify checksum ========================
    msiDataObjChksum(*Dest1, "forceChksum=", *Chksum);
    if(*Check != *Chksum) {
      writeLine("*Lfile", "Checksum failed on *Dest1");
    }
# =====  Delete file from staging area if checksum is good ========
    else {
      msiDataObjUnlink("objPath=*Src1++++forceFlag=", *Status);
    }
  }
}
INPUT *Stage =$"stage", *Coll=$"Archive",
*DestZone=$"tempZone", *Res=$"demoResc"
OUTPUT ruleExecOut
```

Figure 5. Staging Policy

Additional policies are needed to check access controls on files in the archive, verify checksums periodically, verify presence of required metadata, and identify file types. For policies that have been verified to work correctly, the execution of the rule can be automated. Rules can be run periodically, or executed at policy enforcement points. The choice usually depends upon whether batch processing is preferred, or whether continuous processing is needed to manage the workload. The archive administrator has control over the policies that are being applied.

The DFC preservation policy toolkit contains multiple policies for preservation, which can be categorized at an abstract level as:

- Authenticity
- Integrity
- Authorization
- Chain of custody
- Persistent storage management
- Ingestion
- Dissemination
- Fidelity
- Original arrangement and
- Packaging

As was shown with the firewall maintenance policy (which is central to ingestion and dissemination) similar integration of operational procedures had to be done for other abstract policies. In many instances, the technology needed to apply the rule is implemented in an external system. The systems identified in parentheses within the following preservation task list show external services that have been integrated into the DFC environment for providing preservation functionalities. The toolkit contains several snippets of code that can be chained to enable creation of additional policies:

- Automate application of access restrictions.
- Transform data sets to non-proprietary formats.
- Generate event preservation metadata.
- Automate enforcement of user submission agreements.
- Automate creation of checksums.
- Automate capture of description metadata.
- Automate data archiving.
- Automate de-identification of data sets (BitCurator [6]).
- Apply unique identifiers to data (Handle system [7]).
- Enforce authentication of users (InCommon [8]).
- Map metadata terms across ontologies (HIVE [9]).
- Export data in multiple formats (NCSA Polyglot [10]).
- Track usage (Databook).
- Check for viruses (ClamScan [11]).
- Control data retention period.
- Control data disposition.
- Control searches.
- Generate storage cost reports.
- Replicate datasets.
- Copy datasets.
- Synchronize datasets.
- Verify checksum.
- Verify metadata compliance.
- Verify access control against requirements.
- Verify arrangement against requirements.
- Verify format compliance (e.g. XML).

7. COMPARISON WITH ISO 16363

The viability of the DFC preservation approach can be evaluated through comparison with prior preservation audit checklists. Specifically, can each of the tasks defined in prior checklists be turned into computer actionable rules?

An analysis of the ISO 16363 audit checklist has been done to identify which tasks can be automated. The analysis identified 140 preservation tasks. By casting the tasks in terms of generic operations, the number of tasks can be minimized. This requires identifying the state information that will be needed when applying the generic task. An example is a generic rule to print a report. The required state information is the location of the report (logical name) within the preservation environment.

For each task, the predominate operation has been identified, along with the type of entity that is being manipulated. Seven generic operations were defined:

Create, Read, Update, Delete, Copy, Move, & Execute.

The operations were applied to seven object types:

File, Metadata, Events, Policies, Procedures, Database & Ontology.

Examples of the operations upon objects are shown in Table 3. A representative task is selected for inclusion in the list for each combination of operation and object. Thus the "Create" operation can be applied to files, metadata, policies, procedures, events, databases and ontologies. Each task actually may involve multiple operations. Thus an integrity check will verify checksums, delete bad copies, and replace the bad copies from a good replica.

Table 3. Computer actionable task list for ISO 16363

Operation	Object	Task
Copy	file	Create authentic copy from master, verify checksums
Create	Database	New database from metadata in a federated archive
Create	events	Record all micro-services applied to file, along with state information
Create	file	Generate AIP based on AIP template
Create	metadata	Create GUID, handle and logical name for record
Create	ontology	Ontology for designated community terms
Create	policies	Set access policies from remote federation
Create	procedure	Create queries on descriptive metadata
Execute	procedure	Apply transformative migration on format
Move	file	Migrate records to new storage resource
Read	events	List persons who applied archival functions, or accessed file
Read	files	Verify presence of all records specified in submission agreements
Read	metadata	List all persons with access to a collection
Read	policies	List rules for collection
Read	procedure	Verify mechanisms for mitigating risk of data loss
Update	ontology	Remove obsolete terms, incorporate new terms

A second observation is that multiple tasks were required for each criterion specified in the ISO 16363 audit checklist. This raises the question for whether it is possible to identify fundamental criteria that reduce a task to a single operation on a single type of object. Based on this analysis, this will be very difficult to do, since each criterion currently accesses multiple state information attributes to correctly apply a generic operation, interacts with multiple file replicas, and generates multiple event notifications.

The objective of creating computer actionable policies for each task remains a viable approach to preservation. Generic operations can simplify the implementation of preservation tasks while policies can manipulate the multiple objects needed to execute the preservation tasks. This makes it possible to automate preservation processes.

8. SUMMARY

Policy-based data management systems enable creation of preservation environments that maintain records in their original form (persistent objects), while managing interactions with the changing technology in the external world. A preservation environment enables:

- Communication with the future. Records archived today can be retrieved by a future archivist.
- Validation of communication from the past. An archivist can verify the set of policies that governed preservation of a record.
- Management of new technology. A preservation environment allows the flow of technology through the archives while preserving the original records. As new technology becomes available, the technology can be incorporated into the archive without affecting the persistent objects.

Policies are used to enforce assertions that are made about the properties of the preservation environment. Policies are periodically executed to verify the assertions, since storage systems may fail, networks may fail, operators may run obsolete procedures, and software system may malfunction. All assertions made about a preservation environment have to be verified over time. Automating validation of assessment criteria is essential when making assertions such as trustworthiness of a repository.

A generic policy template can be used to define the required policy components. Based on the policy toolkits developed within the DFC, a generic policy template includes:

- Policy name,
- Constraints controlling policy application,
- State information evaluated by the constraints,
- Operations performed by the policy,
- State information needed for operation execution.

With this information, policies can be implemented that automate each preservation task.

9. ACKNOWLEDGMENTS

The development of the iRODS data grid was funded by the NSF OCI-1032732 grant, "SDCI Data Improvement: Improvement and Sustainability of iRODS Data Grid Software for Multi-Disciplinary Community Driven Application," (2010-2013). The results presented in this paper were funded by the NSF Cooperative Agreement OCI-094084, "DataNet Federation Consortium", (2011-2015). The iRODS Consortium developed the pluggable architecture used by the DFC.

10. REFERENCES

[1] Moore, R., A. Rajasekar, "Reproducible Research within the DataNet Federation Consortium", International

Environmental Modeling and Software Society 7th International Congress on Environmental Modeling and Software, San Diego, California, June 2014, http://www.iemss.org/society/index.php/iemss-2014-proceedings.

[2] Rajasekar, A., Wan, M., Moore, R., Schroeder, W., "Micro-Services: A Service-Oriented Paradigm for Scalable, Distributed Data Management", in "Data Intensive Distributed Computing", January 2012, ISBN13: 9781615209712, pp. 74-93.

[3] ISO 16363:2012, "Space data and information transfer systems – Audit and certification of trustworthy digital repositories", http://www.iso.org/iso/catalogue_detail.htm?csnumber=56510.

[4] Moore, R., R. Stotzka, C. Cacciari, P. Benedikt, "Practical Policy Templates", submitted to Research Data Alliance as a deliverable of the Practical Policy Working Group, February, 2015.

[5] iRODS Consortium, http://irods.org.

[6] Lee, Christopher A., Kam Woods, Matthew Kirschenbaum, and Alexandra Chassanoff. "From Bitstreams to Heritage: Putting Digital Forensics into Practice in Collecting Institutions". White Paper. September 30, 2013.

[7] The Handle System, http://www.handle.net/index.html.

[8] InCommon, https://www.incommon.org.

[9] Helping Interdisciplinary Vocabulary Engineering, https://code.google.com/p/hive-mrc/.

[10] NCSA Polyglot, http://isda.ncsa.uiuc.edu/NARA/conversion.html.

[11] ClamAV, http://www.clamav.net/index.html

[12] Harvard Dataverse Network, https://thedata.harvard.edu/dvn/.

[13] Data Observation Network for Earth, https://www.dataone.org.