Outcomes Policy Templates: Practical Policy Working Group, February 2015

Version: February 4, 2015

Abstract

The RDA Practical Policy Working Group was founded in Sept. 2012. The following goals were reached:

- Collection of policies in the RDA Wiki
- Categorization of policies
- Survey of the 11 the most popular policy sets in 30 institutions
- Description of policy templates for these 11 policy sets
- Implementation examples for selected policies and English Language Descriptions

Furthermore an additional Interest Group will be founded to provide various testbeds for all RDA WG/IGs to demonstrate implementations of policy sets and interoperability.

In this document the outcomes of the working group and the templates of the policies are presented.

Table of Contents

1. Introduction

Computer actionable policies are used to enforce management, automate administrative tasks, validate assessment criteria, and automate scientific analyses. The benefits of using policies include minimization of the amount of labor needed to manage a collection, the ability to publish to the users the rules that are being used, and the ability to automate process management.

In cooperation with the Engagement Interest Group of the Research Data Alliance, the Practical Policy Working Group has conducted a survey of production data management systems to elicit the types of policies that are being enforced. The types of data management applications included archives, digital libraries, data grids for data sharing, and processing pipelines. The 30 surveyed sites used more than ten different data management systems, including the integrated Rule Oriented Data System (iRODS), dCache, Tivoli Storage Manager, Xrootd, CLASS, AFS, GPFS, Data Direct Networks Web Object Scalar, Fedora Commons, Dataverse, LOCKS – Lots of Copies Keep Stuff Safe, and XSEDE.

Policy	Importance
Integrity	217
Preservation	150
Access control	126
Provenance	108
Data Management plans	99
Publication	75
Replication	66
Data staging	52
Federation	37
Metadata sharing	23
Regulatory	16
Collection properties	7
Identifiers	7
Data sharing	7
Versioning	7
Licensing	6
Format	6
Data Life Cycle	6
Arrangement	5
Processing	5

Results of a survey of 30 institutions for highest priority policies

Across these diverse environments, the survey identified eleven generic policies that were of interest to a majority of the institutions and are common to almost all data management systems. The policies ranged from management of data access, to control of backups, to provision of contextual metadata:

1. Contextual metadata extraction
2. Data access control
3. Data backup
4. Data format control
5. Data retention
6. Disposition
7. Integrity (including replication)
8. Notification
9. Restricted searching
10. Storage cost reports
11. Use agreements

The organization of a set of policies for a data management system can be expressed as a concept graph that defines relationships between policy concept components. Within this paper, a collection is viewed as an ordered arrangement of digital objects with associated metadata for provenance, description, and administration information. Each collection is assembled for a purpose that reflects the goals of the organizing community. Thus the overriding goal may be the construction of an archive, or digital library, or data sharing environment. The expectation is that the types of operations performed within the data management environment will be similar (discovery, storage, access, manipulation), but the organizing goals may be quite different.

The collection purpose defines the properties that should be maintained for each digital object within the collection. Example properties can include preservation assertions such as authenticity, integrity, chain of custody, and original arrangement; or be based on digital collection assertions such as description and arrangement by subject; or be based on systemic properties of the collection such as completeness, correctness, and consistency.

To enforce a desired property, policies are defined that control the execution of administrative procedures. The application of a policy may be limited, for example, to members of a user group, or files in a collection, or a particular type of data format, or files on a specific storage resource. The limitations are expressed as constraints within the policy. The evaluation of a constraint typically requires access and manipulation of state information (metadata) about digital objects, collections, users, and storage resources.

Each procedure may encompass multiple operations that are chained together into a workflow. An operation may require use of state information that has been stored as metadata on the files in the collection, or metadata stored about storage systems, or metadata stored about users. Each execution of an operation will update the state information, which must be consistently stored and managed.

Policies can be defined that periodically verify assertions about the collection. A common verification policy is analysis of the integrity of the files. This requires checking whether files have been corrupted, and whether the required number of replicas is still available, and whether the replicas are correctly distributed across storage resources.

Figure 1 lists the concept graph that is used to define policy components. Note that the data management environment must provide a way for the policies to be automatically triggered. This may be at a policy enforcement point within the data management infrastructure, or may be an event that is initiated by a user, or may be driven periodically by a timer. If the events are triggered within the middleware infrastructure, it becomes possible to ensure that the policies are applied to control actions by all of the clients that access the system. The policies can be enforced across distributed storage systems, across institutions, and across projects.

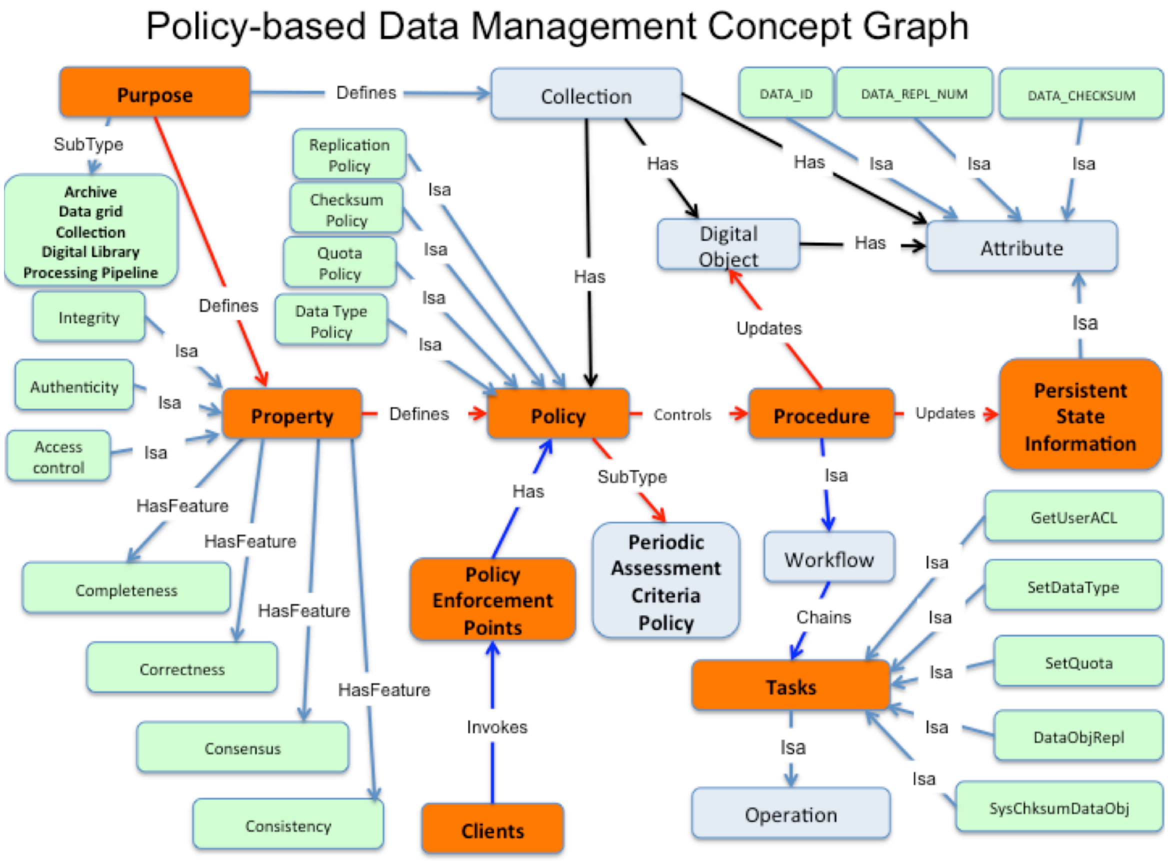

Figure 1. Policy Components

This report provides policy templates for the production policies. Each policy template contains:

- Policy name
- Example constraints that control application of the policy
- State information that is needed to evaluate the constraint
- Example operations that are performed by the policy
- State information that is needed to execute the operations

For all policies, a description of the motivation for the policy is provided. Example policies that implement the management objectives are presented in an additional document "Implementations: Practical Policy Working Group, September 2014".

Each of the generic policy areas actually represents a set of policies. Policies are needed to set environmental variables that control the execution of the policy; to enforce desired collection properties; and to validate assessment criteria.

Each policy example can be modified to implement the specific policy required by an institution. Thus the policies should be treated as examples of approaches for controlling a desired property of a data management system.

2. Contextual metadata extraction policies

The creation of provenance and descriptive metadata defines a context for interpreting the relevance of files in a collection. Depending upon the data source, there are multiple ways to provide metadata:

- Extract metadata from an associated document. An example is the medical imaging format DICOM.
- Extract metadata from a structured document which includes internal metadata. Examples are FITS for astronomy, netCDF, and HDF.
- Extract metadata by parsing patterns within the text within a document.
- Identify a feature present within a file and label the file with the location of the feature that is present within the file.

This policy area focuses on metadata associated with files and collections.

Contextual metadata extraction policy template

The template illustrates types of constraints, the metadata needed to evaluate the constraint, and types of operations that may be applied. Multiple types of metadata may be relevant, including information about provenance, description, data structure, representation, administration, and events. For each type of metadata, there exist standard schemas that may be used to name and organize the metadata. Thus event metadata is typically managed using the PREMIS schema.

Contextual metadata	**Constraint**	**State attributes for Constraint**
for provenance	On file	File_name
for description	On collection	Collection_name
for structuring	On user	User_name
for representation	On storage	Storage_name
for administration	**Operations**	**State Attributes for Operation**
for event	Extract metadata	Attribute_name
		Attribute_value
		Attribute_unit
		Source_file
		Source_collection
	Register metadata	Attribute_name
		Attribute_value
		Attribute_unit
		Destination_file
		Destination_collection
		Metadata_creation_time
		Metadata_modification_time
	Verify metadata load	File_name
		Attribute_name

		Attribute_value
		Attribute_unit
	Verify metadata names	Attribute_name HIVE_reserved_vocabulary
	Set ACL on metadata	File_ID
		Metadata_ID
		ACL_type

3. Data access control policies

Most data management systems provide access controls that limit the ability to modify or add files to a collection, while allowing the public to read public data. The approaches that are taken include:

- Access permission based on the role of the user (all users with a given role are given access to a file or collection)
- Access permission based on an access control list (only named persons or persons with membership in a named group are given access to a file or collection)
- Access permissions that differentiate between read, create, modify, add metadata
- Access permissions that are inherited from the collection

The access controls assume that each user has a unique name, that each file has a unique name, that each collection has a unique name, that each access role has a unique name, and that each access permission has a unique name. In many data management systems, multiple naming conventions may be used. For example, a user may be identified by:

- User_ID, a unique number assigned to the user
- User_name, an ASCII string assigned to the user

The policies may be written based on either the User_ID or the User_name.

An access control can be viewed as a relationship that is established between the unique name for the user, the unique name for a file, and the unique name for the operation performed upon the file. The relationship is stored in an information catalog within the data management system. The operations that are performed by an access control policy will need to include:

- Establishment of unique identities for users, files, collections
- Establishment of an access control on a file or collection
- Validation of the access control for authorizing each type of operation
- Verification of the access controls that are set on files and collections for audits

It may be necessary to include time stamps to log when an access control was set or changed. The information may be stored as metadata on the access control, or as events logged in an audit trail, or as events that are managed in triple store.

Data access control template:

This template includes operations to establish unique names for users, files, collections, and role; operations to set access controls by file or through inheritance from a collection; operations to handle access to replicas; and operations to audit which access controls have been established.

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

The access controls are typically enforced directly by the data management system. However, periodic rules to audit the access controls that have been set are also needed.

4. Data backup policies

A backup corresponds to a copy of a collection that is made at a specific date. A sequence of backups can be made, enabling versions of data files to be tracked over time. Typical state information includes defining the backup time interval, where the backups should be created, and when the backups should be checked.

Data backup policy template:

Backups	**Constraint**	**State attributes for Constraint**
	By time	Periodic time interval
	By collection	Collection_name
	By user	User_ID
	Operations	**State Attributes for Operation**
	Set periodic time interval	Backup_periodicity
	Create file copy	File_name
		Storage_repository_name
		Backup_collection_name
	Set backup time stamp on file	File_time_stamp
	Verify backups	File_time_stamp

5. Data format control policies

Many collections restrict the types of data formats that will be acceptable for ingestion. Policies that identify data formats that are not allowed can either send warning messages, or move the file to a staging area, or attempt to transform the data format. Policies can be written to manage staging areas based on the type of data format, sorting selected data formats into specified collections. The associated operations include creating metadata to list the file format type, checking file formats, and verifying file formats.

A list of accepted formats can be chosen from a Data Type Registry that is built up by the RDA Working Group “Data Type Registries”.

Data format control policy template

Format Requirements	**Constraint**	**State attributes for Constraint**
	On ingestion of file	
	Periodic check	Time interval between checks
	For specific format type	File-format_type
	For collection	Collection_name
	Operations	**State Attributes for Operation**
	Set file format	File_ID
		File_format_type
	Get file format	File_ID
		File_format_type
	Check file format	File_ID
		File_format_type
	Convert file format	File_ID
		File_format_type
		Desired_file_format_type
	Verify file format	Collection_name
		File_name
		File_format_type
		Desired_file_format_type

6. Data retention policies

There are multiple types of retention that may be controlled by policies:

- Retention based on a data expiration date. The expiration date is checked to verify whether a file is a candidate for application of a disposition policy
- Cache management based on the age of files. The oldest files are removed from the cache to make room for new files.
- Retention based on migration. Files in a staging area are moved to an archive after passing quality assurance tests.

Data retention policy template

Data retention	**Constraint**	**State attributes for Constraint**
	By file	File_name
	By collection	Collection_name
	By storage system	Storage_name
	Operations	**State Attributes for Operation**
	Set retention period	File_name
		Retention_period
	Check retention period	File_name
		Retention_period
	Verify retention period	File_name
		Retention_period

7. Disposition policies

Once files have been identified that have exceeded a retention period, a disposition policy can be applied to either delete or archive the files. For the above example for a data expiration policy, a disposition policy can be created that migrates the expired files to an archive collection, or that deletes the expired files.

Disposition policy template

Disposition	**Constraint**	**State attributes for Constraint**
	By file	File_name
	By collection	Collection_name
	Operations	**State Attributes for Operation**
	Define disposition policy	Disposition_policy_type
	Set disposition policy	File_name
		Collection_name
		Disposition_policy_type
	Apply disposition	File_name
		Collection_name
		Disposition_policy_type
		Audit_trail
		Event_log
	Verify disposition	File_name
		Collection_name
		Disposition_policy_type
		Event_log

8. Integrity policies

The management of integrity may require multiple independent steps that are applied to each file:

- Verification of integrity on ingestion through validation of a checksum
- Replication of the file across multiple storage locations to ensure the ability to replace a corrupted file.
- Periodic verification that the files are not corrupted, that the required number of replicas exist, and that the replicas are correctly distributed.

Administrative information is saved for each file to track the number of replicas, the places where the replicas are stored, and the checksums for the files.

Integrity policy template

The choice of how integrity is ensured or verified can be done automatically through Policy-Enforcment-Points, or periodically as an administrative policy.

Integrity	**Constraint**	**State attributes for Constraint**
	Periodic check	Time_interval
	For collection	Collection_name
	For file	File_name
	For storage system	Storage_system_name
	Operations	**State Attributes for Operation**
	Set checksum	File_ID
		File_checksum
	Create replica	File_ID
		File_replica_number
		File_replica_location
		File_replica_creation_time
		File_physical_path_name
		File_replica_checksum
	Verify checksum	File_ID
		File_checksum
	Verify replicas	File_id
		File_replica_number
		File_replica_checksum
		File_storage_location

The administrative information (state attributes) must be consistently managed. When a file is moved to a new storage system, the administrative information must be updated correctly. Note that on ingestion, a manifest, such as Bagit, may list the checksum for each file that is ingested. This manifest could be parsed, with the checksum extracted and stored for each file. This ensures that the file was not corrupted during the initial transfer.

9. Notification policies

There are multiple types of notification that may be triggered by an event. Examples include:

- E-mail to an administrator
- Message sent to an external message queue for processing
- Twitter post

The types of events that are tracked may include:

- Creation of a new collection
- Change of access control permission on a collection
- Deposition of a file into a collection
- Deletion of a file
- Creation of a new user account
- Deletion of a user account
- Federation with another data grid

Notification policy template

Notification	**Constraint**	**State attributes for Constraint**
	For specific event type	Notification_event_type
	For user	User_ID
	For collection	Collection_name
	For file type	File_type
	For file	File_name
	For storage system	Storage_system_name
	Operations	**State Attributes for Operation**
	Set event notification type	Notification_event_type
		Notification_type
		Notification_type_address
	Event notification	Notification_event_type
		Notification_type
		Notification_type_address
		Event_log
	Search events	Notification_event_type
		Event_log

10. Restricted searching policies

Restricted searching can be viewed as a form of restricted access control. A typical example is restricting the ability of users to see any files except their own files. This limits browsing to the viewing of files for which they have read access. Other types of restricted searching include:

- Limiting the ability to search metadata to only those files for which the user has read access.
- Limiting the ability to search metadata to a subset of available metadata for a file. In effect, specific metadata attributes may be restricted to administrator access across all files within the system.
- Limiting the ability to search system level metadata, such as internal function mapping attributes.

Restricted search policy template

Restricted searching	**Constraint**	**State attributes for Constraint**
	By collection	Collection_name
	By user group	User_group_ID
	By user	User_name
	By collection property	Collection_name
		Collection_metadata_name
	By file property	File-name
		File_metadata_name
	By event	Event_type
	Operations	**State Attributes for Operation**
	Issue query	Metadata_name

11. Storage cost policies

The ability to generate usage reports and the associated storage cost is needed for both planning and accounting. The approach can be integrated with the use of quotas to limit the maximum allowed storage usage, which in turn limits the maximum cost.

Storage cost policy template

Storage cost tracking	**Constraint**	**State attributes for Constraint**
	By storage system	Storage_system_name
	By user group	User_group_ID
	By user	User_name
	By collection	Collection_name
	Operations	**State Attributes for Operation**
	Record usage	Storage_repository_name
		User_name
		User_usage
	Audit usage	Storage_repository_name
		User_name
		User_usage
	Generate storage cost report	Report_name

12. Use agreement policies

Not all policies can be implemented as computer actionable rules. The creation of a use agreement policy is typically negotiated at the time an account is established for a user, and involves the receipt of a signed document. It is possible to associate an attribute with each user that defines whether or not the signed use agreement has been received.

Use agreement policy template

Signing of use agreements	**Constraint**	**State attributes for Constraint**
	Operations	**State Attributes for Operation**
	Store agreement	User_name
		Creation_date
		Agreement_name
		Agreement_type
	Audit agreement	User_name
		Creation_date
		Agreement_name
		Agreement_type