Fedora and the Preservation of University Records Project

3.2 Checklist of Fedora's Ability to Support Maintain Activities

Version
1.0

Date
September 2006

Digital Collections and Archives, Tufts University
Manuscripts & Archives, Yale University

An Electronic Record Research Grant funded by the
National Historical Publications and Records Commission

Digital Collections and Archives
Tisch Library Building
Tufts University
Medford, Massachusetts 02155
http://dca.tufts.edu

Manuscripts and Archives
Yale University Library
Yale University
P.O. Box 208240
New Haven, Connecticut 06520-8240
http://www.library.yale.edu/mssa/

Co-Principle Investigators
Kevin Glick, Yale University
Eliot Wilczek, Tufts University

Project Analyst
Robert Dockins, Tufts University

This document is available online at
http://dl.tufts.edu/view_pdf.jsp?urn=tufts:central:dca:UA069:UA069.004.001.00010
(September 2006)

Fedora and the Preservation of University Records Project Website at
http://dca.tufts.edu/features/nhprc/index.html

Funded by the
National Historical Publications and Records Commission
Grant Number 2004-083

Fedora and the Preservation of University Records Project

TABLE OF CONTENTS

OVERVIEW

This Checklist builds directly on the Maintain Guide, describing the abstract services that support the prescribed responses to the set of scheduled and irregular event types presented by the Guide. The Checklist describes services in the most generic and abstract sense; this notion of service is intended to include "manual" services performed by people, "automated" services implemented in software, as well as various combinations of the two. Some services are most easily implemented as manual processes, some as software services, and others can be reasonably implemented in a variety of ways.

Understanding the Checklist depends significantly on reading the Maintain Guide. The Guide presents a set of scheduled and irregular event types that support various Maintain steps. The event types are non-sequential. Each event type is triggered by various circumstances or conditions. The Guide describes the nature of each event type, the conditions for their occurrences, and a set of steps an Archive (or its designee) needs to undertake when an event type occurs. An Archive will need to utilize at least some of the abstract services the Checklist describes in order to undertake the steps set out by the Maintain Guide in a scaleable manner.

The Checklist also analyzes Fedora's ability to support the abstract services identified. This analysis examines if Fedora currently utilizes the abstract services as part of its core architecture, in its service framework, or as an external service that it communicates with. The Checklist discusses current and potential future development work for abstract services that do not currently work with Fedora in some capacity. This future development work often refers to the activities of the Fedora Preservation Services Working Group, a body within the Fedora community charged to develop a "general definition of preservation services for the Fedora service framework [and]… recommend enhancements to the [core] Fedora repository service as well [as] develop specifications for new preservation-support services for the Fedora Service Framework."[1] This Checklist measures its analysis of Fedora against version 2.1 (released February 2006).

Form of the Checklist

After providing an overview and a description of Fedora the Checklist lists nineteen abstract services, providing for each service a description, an analysis of Fedora's support for the service, and a list of event type steps described in the Maintain Guide that the service supports. In Appendix A, the Checklist has a crosswalk of the Maintain Guide event type steps to the abstract services that support the steps.

Fedora

Fedora is open repository application.[2] Rather than an out-of-box, limited repository solution, Fedora is a repository architecture upon which an institution can shape a repository in many different ways. It is a developer-oriented, rather than user-oriented, application that allows repository managers to build (or use existing) services on top of Fedora, including ones such as search services that interface with users.

[1] "Working Group: Preservation," <http://www.fedora.info/wiki/index.php/Working_Group:_Preservation>
[2] Fedora stands for Flexible Extensible Digital Object Repository Architecture. For more information on Fedora see <http://www.fedora.info>

Fedora is essentially a set of repository services. At its kernel, are the services that formulate the Fedora Core that is surrounded by services in the Fedora Service Framework. Services in the Core and the Framework also work with external services that reside outside a Fedora repository. The Fedora Core[3] is a core repository application that ingests digital objects and its data-streams; validates the object wrapper (usually FOXML) and the presence of declared data-streams; stores (internally or externally), registers, indexes, and manages the objects and data-streams; enables search of the objects and dissemination of the data-streams; and enforces access polices. These functions are exposed through four web service APIs:

1) Repository management interface (API-M) that provides write access to the repository
2) Repository access interface (API-A) that provides read access to the repository
3) Basic repository search index
4) RDF-based search of the Core's resource index.

The Fedora Service Framework allows for a more modular preservation system, because services are implemented on top of the Fedora Core, allowing their configuration or replacement with another service.[4] The Fedora Service Framework, which was implemented with version 2.1 (released February 2006), surrounds the Fedora Core and allows the smooth integration of the Core with new services. These services operate independently of the core repository application. Managers of Fedora instances are free to choose which services to implement. This keeps Fedora agile and better able to respond to emerging needs by extending its services while leveraging existing resources and avoiding the need to do costly and risky core overhauls.[5]

If configured properly, a Fedora repository instance can communicate with external services. For example, an external workflow system can communicate with a specialized ingest service in the Framework to undertake batch ingests. An integrity service could work with an external service such as JHOVE to validate the format of data-streams ingested into a repository. This allows Fedora to extend its services further by leveraging the work of resources completely outside the repository.[6]

[3] In this document, Fedora Core refers to the core services for the Flexible Extensible Digital Object Repository Architecture (Fedora) developed by Cornell University and University of Virginia Library. See <http://www.fedora.info> and <http://www.fedora.info/wiki/index.php/Fedora_Core_Repository_Service>. This does *not* have anything to do with Red Hat releases of their community-oriented Linux distribution.
[4] "Fedora Core Repository Service," <http://www.fedora.info/wiki/index.php/Fedora_Core_Repository_Service>
[5] "Fedora Service Framework," <http://www.fedora.info/wiki/index.php/Fedora_Service_Framework>
[6] Ibid.

ABSTRACT SERVICES

AIP Module

Description

This module helps the Archive manage Archival Information Packages (AIPs) within its Preservation System. AIPs consist of Content Information, Preservation Description Information, and Packaging Information.[7] The module facilitates the semi-automated management of AIPs. For example, when the Archive transforms the format of thousands of records to a new format, the AIP Module facilitates the addition of the new bitstreams to the AIPs.

Fedora Support

Fedora manages AIPs as part of its *Core* services, undertaking the creation, modification, and deletion of digital objects and its data-streams, executing the activities of ingest, validation, management, storage, register, index, search, and disseminate. Fedora exposes these activities through its Fedora Management service, known as API-M Methods. In addition, Preservation System managers might utilize management tools, like FEZ and ELATED. Currently these services provide some of the management functionality needed in an AIP Module.[8] The API-M Methods, FEZ, or ELATED cannot support all the specialized needs of an AIP Module. For example, they are not explicitly designed to address the discovery of all records components "orphaned" by lost AIP(s) (Respond to Data Loss: Administrative Metadata Store) or the ability to generate a new "record fragment" AIP for each orphaned record component (Respond to Data Loss: Administrative Metadata Store).[9]

Supports Event Type Steps

Format Transform Request: Step 6
Remove Record Component Request: Step 2
Preservation Application Hardware Environment Replacement: Step 9
New AIP Format and/or New Preservation Application: Step 9
Respond to Data Loss: Administrative Metadata Store: Steps 1, 2

Alerting Service

Description

A service that alerts a repository manager or service that an event has occurred that may require the execution of a subsequent Maintain activity. The service provides the infrastructure for communicating event occurrences to the proper services or managers. For example if Storage Management Module generates a report on the age of storage media, it uses the Alerting Service to send that report to Administration. Archive Administration and Preservation Planning could configure an Alerting Service to feed

[7] ISO 14721:2003*:* Space data and information transfer systems--Open archival information system--Reference model (Geneva: International Organization for Standardization, 2003).
[8] "FEZ development at the University of Queensland Library," <http://www.library.uq.edu.au/escholarship/fezdev.pdf> and "Elated: a general-purpose web-based client for the Repository System," < http://elated.sourceforge.net/ >. Also see "Tools," <http://www.fedora.info/tools/index.shtml>.
[9] Administrators of a Fedora-based Preservation System may use a variety of techniques to discover "orphaned" records components or generate "records fragments" but modules specifically designed to undertake these tasks do not exit.

them reports on appropriate Maintain activities to help ensure that they are properly monitoring the records in the Preservation System and can respond to events in a timely manner.

Fedora Support

Fedora *Core* does not include an Alerting Service, nor does one currently exist as a *Framework* resource. While tools exist that can execute as least some of the Alerting Service function, no one has utilized them with Fedora to date. The Fedora Preservation Services Working Group is currently engaged in developing an Alerting Service that would be part of the Framework and may require some small adjustments to the Core. The Fedora Development Team has begun work on message-enabling the *Core* and the *Service Framework*, which lay the architectural foundation for the Alerting Service. The Working Group envisions the Alerting Service as part of a larger Event Management Service that documents, encodes, and facilitates management of events that occur within the repository. The Alerting Service may in future serve as the underlying communication architecture between a broad range of services in the Fedora *Framework* and *Core*.[10]

Supports Event Type Steps

Check Access and Retention Status: Step 2
Report on Media Life: Step 2
Hardware Test and Maintenance Window: Steps 4, 6
Digital Object Accession: Step 7
Format Transformation Request: Step 1
New AIP Format and/or New Preservation Application: Step 9
Change Standard Computing Platform: Step 3
Respond to Checksum Failure: Step 2
Respond to Media Failure: Record Component Store: Step 6
Respond to Data Loss: Record Component Store: Step 1
Respond to Media Failure: Administrative Metadata Store: Step 9
Respond to Unintentional Data Damage: Step 3
Respond to Security Breach: Step 6

Data Backup Protocol

Description A set of rules and procedures for administering a data backup process. This usually includes defining backup types (full or incremental), the frequency of backups, the storage of backup data, the cycling and retention of backup data, and the responsibilities for administering the data backup process. The Data Backup Protocol is closely related to the Storage Management Module and may be a subset of that Module.

Fedora Support

Currently, neither Fedora *Core* nor the *Framework* manages data backup processes, nor does an *External* data backup service built specifically to work with Fedora exist. Managing a Data Backup Protocol completely separate from Fedora does not cause problems per se, but it would be helpful to connect data backup administration with the digital objects in the repository. For example, retention and disposition metadata may have a significant impact on the cycling and retention of backup media.

Supports Event Type Steps

Incremental Backup of Administrative Metadata: Steps 1, 2

[10] Fedora Group: Preservation <http://www.fedora.info/wiki/index.php/Working_Group:_Preservation>.

Full Backup of Administrative Metadata: Steps 1, 2, 3
Incremental Backup of Records Component Store: Steps 1, 2
Full Backup of Records Component Store: Steps 1, 2
Preservation Application Hardware Environment Replacement: Step 12
New AIP Format and/or New Preservation Application: Step 12
Respond to Checksum Failure: Steps 3, 4, 5, 6, 7
Respond to Media Failure: Record Component Store: Step 2

Data Management Database

Description

This service corresponds closely to the OAIS data management functional entity. It is primarily responsible for enabling the discovery of objects in the repository. As such, it usually takes the form of some manner of a database that supports a query language. This service differs from the search service in that the search service is concerned with the execution of queries and presentation of results, whereas the Data Management Database is concerned with maintaining the data stores which supports such queries.

Fedora Support

Fedora 2.1 comes with two mechanisms for discovering objects in the repository. The first mechanism is called "Basic Search." It is a simple text-based search of the basic properties of the FOXML that wrap Fedora objects and the Dublin Core metadata. The basic search is very simplistic.[11] The second native mechanism for discovering Fedora objects is the RDF triple store. Every Fedora object can have a distinguished metadata stream which contains RDF statements. These RDF statements are maintained in a triple-store which supports arbitrary iTQL queries. It is additionally possible to maintain external search indexes. However, currently maintaining such an external index requires either 1) using an external application which manages both ingest and the index or 2) keeping the index in sync with the repository manually. However, an Alerting Service (see below) would allow one to automatically maintain such an external index.

Support Event Type Steps

Digital Object Accession: Step 6
Metadata Update Request: Step 4
Format Transformation Request: Step 8
Remove Record Component Request: Step 4
Preservation Application Hardware Environment Replacement: Step 8
New AIP Format and/or New Preservation Application: Step 8

Format Transformation Service

Description

This service performs formats transformations of records components, moving a record component from one file format environment to another. This service should be able to undertake batch transformations.

Fedora Support

[11] The relational database underlying a Fedora repository has considerably more information in it than is exposed by the Basic Search interface. Most notably, Basic Search does not show all of the registry and storage tables, plus other control tables. Preservation System administrators can make standard SQL queries directly on these tables. However, Fedora does not expose this information via a public API.

Fedora *Core* does not include a Transformation Service at the individual records component level. While Preservation System administrators can use the Fedora API-M to modify a digital object metadata stream while the digital object remains in place in the repository, it is not a transformation service that transforms data-stream formats—from a MS Word file to a PDF/A file, for example.[12] The Fedora Preservation Services Working Group is considering the possibilities of adopting some kind of Transformation Service to work with Fedora, most likely as an *External* service. Preservation System managers should be able to successfully manage their own Format Transformation Services that only communicate with their Fedora repositories as *External* services.

Supports Event Type Steps

Format Transformation Request: Step 2

Format Validation Service

Description

This service checks and validates the file formats of digital objects. JHOVE is a prominent example of a Format Validation Service. Ideally, a Preservation System should have a Format Validation Service that can validate all of its Preservation Formats and formats it ingests. The Service should also be able to batch-validate. Thus, a Validation Service should have the ability to add new formats to its validating repertoire.

Fedora Support

While Fedora *Core* does validate the digital object wrapper (usually FOXML files) and can confirm the existence of all declared data-streams, it does not include a Validation Service for individual records components. For example, while it can confirm the existence of a data-stream that belongs to a digital object in a Fedora repository, it cannot confirm the format of that data-stream—determining, for example, if a file really is a PDF-A file as it claims. The Fedora Preservation Services Working Group is examining the possibility of developing a Validation Service within a broader preservation integrity service that would call on External validations services such as JHOVE or GDFR. While system managers can successfully run this service completely separate from Fedora, it would help managers monitor the long-term integrity of the digital objects in their repositories.

Supports Event Type Steps

Format Transformation Request: Step 3

Integrity Checking Protocol

Description

This protocol is a set of rules and procedures that defines the methods, timing, and responsibilities for checking the integrity of records components in a Preservation System. In defining the methods, the Protocol incorporates an integrity checking service(s) that are composed of an integrity checking tool(s), such as checksums.

Fedora Support

At the moment, the Fedora *Core* does not come packaged with an Integrity Checking Protocol or tools for checking the integrity of data-streams or digital objects. However, checksum for datastreams is already specified in the Fedora data model (FOXML). Currently, the Fedora Development Team is working on enabling support for checksums

[12] An administrator of a Fedora-based Preservation System could currently build a disseminator to transform data-streams in MS Word to a PDF/A formatted Dissemination Information Package.

in *Core*. This would allow a Preservation System administrator to configure the repository core of his or her Preservation System to automatically calculate checksums for datastreams as part of completing any API-M transaction. The Development Team is also creating a new Fedora API-M method that will allow clients to make requests to (1) calculate and store the checksum of a datastream on demand, and (2) run a comparison of a checksum to the current state of datastream content to see if there is a change. The Development Team expects to complete this work by the fourth quarter of 2006. A manager of a Fedora-based Preservation System can successfully manage an Integrity Checking Protocol that is external to the Fedora repository. Fedora will never define the protocol of the methods, timing, and responsibilities of the integrity checking—that is something that Preservation System managers will have to develop on their own or adopt from external integrity checking standards or best practices. The Fedora Preservation Services Working Group is currently investigating integrity checking tools as a *Framework* service that will allow managers to more easily report on an integrity checking event in order to document the occurrence of a check—and the results of that check—in digital objects' metadata or repository log files. This work is closely related to the Working Group's development of its Alerting Service.

Supports Event Type Steps

Verify AIP Consistency: Step 1
Verify Records Components: Step 1
Respond to AIP Consistency Failure: Step 2
Respond to Unintentional Data Damage: Step 2
Respond to Security Breach: Step 3

Knowledge Base Module

Description

This module facilitates an Archive's tracking and defining a Designated Community's Knowledge Base. The Knowledge Base of a Designated Community is the Community's language ability, subject knowledge, and capabilities of its Standard Computing Platform that it needs to understand records in a Preservation System. Archives attach Representation Information to records to ensure that members of its Designated Community can understand those records with their Knowledge Base. As a Designated Community's Knowledge Base changes over time, an Archive must adjust its Representation Information accordingly.[13]

Fedora Support

Neither Fedora *Core* nor the Service *Framework* includes a Knowledge Base Module and there is no external module that Fedora connects to either. The concept of Knowledge Base has not matured greatly in the digital preservation community so no such module yet exists. Fedora could use its de facto semantic web capabilities, as manifested in the Resource Index, to establish a relationship between objects in a repository and the technical capabilities of the Designated Community to view the objects. Fedora's innate RDF support provides significant flexibility for expressing a wide variety of objects and their relationships, such as object properties and their relationships to external entities like the Knowledge Base of a Designated Community. The Fedora Development Team has started work on providing a more robust and simpler triplestore capabilities in the

[13] According to ISO 14721:2003, a Knowledge Base is, "a set of information, incorporated by a person or system, that allows that person or system to understand received information.

Core and Service *Framework*, which would give Preservation System administrators a more robust and flexible environment for characterizing, tracking, and modifying the Knowledgebase Base of a Designated Community. However, this does not provide direct guidance on how to define and characterize the Knowledge Base of a Designated Community over time.

Supports Event Type Steps

Change Standard Computing Platform: Step 2

PDI Module

Description

This module helps the Archive manage Preservation Description Information (PDI) which is a component of every record (AIPs) within its Preservation System. PDI is the provenance, reference, fixity, and context information that is needed to preserve the records in a Preservation System.[14] The module facilitates the semi-automated management of PDI. For example, when the Archive ingests thousands of records (SIPs) and turns them into thousands of AIPs, this module helps to generate the necessary PDI to be part of those AIPs. The PDI Module is closely related to the AIP Module and in some cases could be a component of the AIP Module.

Fedora Support

Just as with AIPs, Fedora essentially manages PDI as part of its *Core* services, undertaking the creation, modification, and deletion of digital object metadata. Much of Fedora's ability to support the proper administration of PDI will depend on a Preservation System manager's configuration of metadata. This metadata will need to properly capture and manage the appropriate provenance, reference, fixity, and contextual information.

Supports Event Type Steps

Verify Records Components: Steps 1, 3
Check Access and Retention Status: Step 3
Hardware Test and Maintenance Window: Steps 6, 7
Security Audit: Step 3
Digital Object Accession: Step 4
Metadata Update Request: Step 2, 3
Format Transformation Request: Step 7
Remove Record Component Request: Step 3
Preservation Application Hardware Environment Replacement: Steps 6, 10, 11
New AIP Format and/or New Preservation Application: Steps 6, 10, 11
Refresh Records Component Media: Step 6
Respond to Checksum Failure: Steps 1, 8
Respond to Media Failure: Record Component Store: Steps 1, 5
Respond to Data Loss: Record Component Store: Step 3
Respond to Media Failure: Administrative Metadata Store: Step 1
Respond to Data Loss: Administrative Metadata Store: Steps 3, 4
Respond to Unintentional Data Damage: Step 5
Respond to Security Breach: Step 4, 9

[14]ISO 14721:2003, Space data and information transfer systems – Open archival information system – Reference model.

Persistent Identifier Manager

Description

This resource assigns persistent identifiers to records ingested into the Preservation System and manages these identities over time, ensuring they persist and do not conflict with other digital objects.

Fedora Support

Through the API-M Methods the Fedora services allow managers to assign persistent identifiers to objects during ingest into the repository. Fedora also allows for the management of those identifiers over time. Assigning and managing persistent identifiers is a *Core* service. However, it is important to note that Fedora does not manage persistent identifiers across multiple repositories. For example, Fedora does not resolve potential conflicts resulting from duplicate unique identifiers found in a federation of Fedora repositories.

Supports Event Type Steps

Digital Object Accession: Steps 1, 3
Format Transformation Request: Steps 1, 4

Repository History

Description

This is a service that generates a log of events in the life of the repository core of a Preservation System. These events may include significant changes in repository hardware or software, changes in administration, or data loss events. They would not include regular activities such as adding new digital objects to the repository core. Ideally, the log would exist as a special digital object associated with the repository and as data-streams that are part of each digital object in the Preservation System, mostly likely service as a part of the provenance information of an object's PDI.

Fedora Support

Fedora *Core* does not include a Repository History, nor does one currently exist as a *Framework* resource. While the manager of a Fedora-based Preservation System can manually create logs of significant repository events, a service for automatically capturing and logging this information does not exit. The Fedora Preservation Services Working Group is examining the feasibility of developing such a service.

Supports Event Type Steps

Hardware Test and Maintenance Window: Step 7
Security Audit: Step 3
Respond to Data Loss: Record Component Store: Step 2
Respond to Media Failure: Administrative Metadata Store: Step 8
Respond to Security Breach: Step 9

Repository Stasis

Description

This is a service that allows a Preservation System administrator to prevent all changes to a System's repository core—both to the repository itself and to the digital objects it manages. In short, stasis freezes the repository. This would also manage who has permission to put a repository core in stasis and take it out of stasis.

Fedora Support

Fedora does not have a repository stasis module as part of its *Core* functionality *per se*.

However, the Fedora XACML policy module can act as a *defacto* stasis module, allowing Fedora to prevent modifications. A Preservation System administrator can express policies in a variety of ways to halt changes, such as rejecting all calls to API-M methods. Naturally, an administrator can also shut down the Fedora application entirely; however, this prevents access in addition to modification, which may not be acceptable in some cases.

Supports Event Type Steps

Hardware Test and Maintenance Window, Steps 1
Preservation Application Hardware Environment Replacement: Steps 5, 13
New AIP Format and/or New Preservation Application: Steps 5, 13
Respond to AIP Consistency Failure: Step 1
Respond to Media Failure: Administrative Metadata Store: Step 1
Respond to Unintentional Data Damage: Step 1
Respond to Security Breach: Step 1

Request Service Manager

Description

This service manages a request for a records component in a Preservation System with the retrieval of the appropriate records component and delivery of the component to the requestor. The service should also generate statistical data on the requests it receives and services.

Fedora Support

Fedora supports this service in its *Core* with its Dissemination function. Currently the Core does not automatically generate statistical data on requests and disseminations, but such functionality is currently being examined as part of the research into the Alerting Service being undertaken by the Fedora Preservation Services Working Group.

Supports Event Type Steps

Retrieval Request: Steps 1, 2, 3, 4

Retention and Disposition Module

Description

This service manages the retention and disposition of records in a Preservation System—determining how long records must be kept in the System and what should ultimately happen to the records. In managing the retention period of records, the module should be able to alert Preservation System administrators of records with retention periods about to expire. In managing records disposition, the module should be able locate and execute the disposition—which is often to destroy—of all the components of the appropriate records. It should also be able to provide administrators with reports on disposition activities.

Fedora Support

Fedora does not currently support the definition of record types with retention and disposition rules as part of its *Core*, *Framework*, or *External* services. However, the Fedora Community may be able to articulate record types as content models, building on the community's current work to formalize content model rules.

Supports Event Type Steps

Check Access and Retention Status: Steps 1, 2
Remove Record Component Request: Step 1

Search Service

Description

This service conducts searches against the metadata of records in the Preservation System and provides query results to the requestor. This service should appropriately limit result sets to the requestor's permissions. In addition, the service should also be able to generate statistical data on the queries it conducts.

Fedora Support

Fedora supports this service in its *Core* with Registry, Access, and ResourceIndex functions within the *Core*. The Fedora Search Working Group has recently developed the Generic Search Service and made it available for downloading. [15] The Fedora Development Team has integrated the Generic Search Service into the official Fedora CVS repository and the service will be part of the Fedora 2.2 release as part of the Service Framework in the fourth quarter of 2006.

Supports Event Type Steps

Query Request: Steps 1, 2, 3, 4

Security Audit

Description

This is a set of rules and procedures for systematically checking the effectiveness and reliability of the Security Protocol. Ideally, the Security Audit should be conducted by entities that do not manage or administer the Security Protocol.

Fedora Support

The security audit rules, procedures, and processes are usually managed externally from the Preservation System and repository core it is auditing. Therefore, there is no pressing need to have the audit be a Fedora *Core* or *Framework* service.

Supports Event Type Steps

Security Audit: Steps 1, 2, 3

Security Protocol

Description

This is a set of rules and procedures for protecting the Preservation System from security breaches, attacks, and unauthorized access. A Security Protocol usually incorporates a wide range of measures from network security to physically securing workstations. A Security Protocol also manages user access to records in a Preservation System.

Fedora Support

Fedora can manage access permissions and rules as part of its Core service employing XACML metadata. XACML is an XML-based language that enables a wide variety of access control and security policies to be expressed. The Fedora XACML enforcement module will enforce the rules expressed in such policies. Fedora started utilizing XACML with version 2.1. Fedora also has support for user authentication, via plug-in Tomcat modules, and it also has support for Secure Sockets Layer (SSL) so it can transmit data-streams over a network securely. Network and workstation security activities naturally fall outside the purview of Fedora.

Supports Event Type Steps

[15] "Working Group: Search," <http://www.fedora.info/wiki/index.php/Working_Group:_Search>. Also see "Fedora Generic Search Service Development," <http://defxws2006.cvt.dk/fedoragsearch/>, which has a prototype of the service available for downloading.

Security Audit: Steps 1, 2, 3
Respond to Security Breach: Step 7

Storage Management Module

Description

A module that manages the storage of records components that are in and managed by the repository core of the Preservation System. The module should be able to track and report where records components are stored, the type and age of storage media, and test storage media performance. The module should facilitate a Preservation System administrator's efficient storage of records components. The Storage Management Module is closely related to and can encompass the Data Backup Protocol.

Fedora Support

Fedora is deployed by default with a file system storage module as part of its *Core.* This default module manages the simple storage of data-streams but does not track and manage the type, age, and performance of storage media, and it turn does not manage which data-streams should be stored on which media. However, Fedora addresses this limitation by having its storage module written with a generic interface, allowing it to plug into different backend storage management systems. This allows Preservation System administrators to replace the default storage module with alternatives, such as commercial hierarchical storage systems. Fedora has released a beta version of a plug-in for the Storage Resource Broker (SRB).

Supports Event Type Steps

Report on Media Life: Steps 1, 2
Hardware Test and Maintenance Window: Steps 1, 5
Digital Object Accession: Steps 1, 5
Metadata Update Request: Step 1
Format Transformation Request: Step 5
Remove Record Component Request: Step 1
Preservation Application Hardware Environment Replacement: Step 7
New AIP Format and/or New Preservation Application: Step 7
Refresh Records Component Media: Steps 1, 2, 3, 4, 5, 7, 8, 9
Respond to Media Failure: Record Component Store: Steps 3, 4

System Administration Protocol

Description

A set of rules and procedures for managing the hardware and networks that support the repository core of the Preservation System. This protocol would usually mirror most standard system administration rules and procedures. The Administration Protocol may also include a preservation capabilities reporting function that reports to administrators the current capabilities of the repository, including storage capacity and system performance, among others.

Fedora Support

Fedora does not offer this Protocol as a *Core, Framework,* or *External* service. A System Administration Protocol is not a natural fit as a Fedora service and will probably always exist separately from Fedora.

Support Event Type Steps

Hardware Test and Maintenance Window: Step 3
Preservation Application Hardware Environment Replacement: Steps 1, 2, 3, 4, 14, 15

New AIP Format and/or New Preservation Application: Steps 1, 2, 3, 4, 14, 15
New Records Component Store: Steps 1, 2, 3, 4
Respond to Media Failure: Administrative Metadata Store: Steps 2, 3, 4, 5, 6
Respond to Unintentional Data Damage: Step 2

APPENDIX A: EVENT TYPE STEPS TO ABSTRACT SERVICES CROSSWALK

Scheduled Event Types

Incremental Backup of Administrative Metadata

Steps	Abstract Services
1. Perform backup of administrative metadata	Data Backup Protocol
2. Store backup data in a secure location	Data Backup Protocol

Full Backup of Administrative Metadata

Steps	Abstract Services
1. Perform full backup image of Administrative Metadata	Data Backup Protocol
2. Move full backup image to secure off-site location	Data Backup Protocol
3. Retain at least one year's worth of full backup images at the secure off-site location	Data Backup Protocol

Incremental Backup of Records Component Store

Steps	Abstract Services
1. Perform backup of records components	Data Backup Protocol
2. Store backup data in a secure location	Data Backup Protocol

Full Backup of Records Component Store

Steps	Abstract Services
1. Perform a full backup image of the Records Component Store	Data Backup Protocol
2. Move full backup image to secure off-site location	Data Backup Protocol

Verify AIP Consistency

Steps	Abstract Services
1. Verify the internal consistency of each AIP	Integrity Checking Protocol
2. If any AIP fails the consistency check, go to **Respond to AIP Consistency Failure**	*See **Respond to AIP Consistency Failure***

Verify Records Components

Steps	Abstract Services
1. Compute checksums on records components and compare to checksums stored in PDI	PDI Module; Integrity Checking Protocol
2. If any checksums are not correct, go to **Respond to Checksum Failure**	*See **Respond to Checksum Failure***
3. Document "Verify Checksum" event in PDI	PDI Module

Check Access and Retention Status

Steps	Abstract Services
1. Identify all records governed by time-based expiration	Retention and Disposition Module
2. Report all such records to Administration	Retention and Disposition Module; Alerting Service
3. Document "Retention or Access Period Expired" event in PDI	PDI Module

Report on Media Life

Steps	Abstract Services
1. Generate a report listing the types and age of all media in the Records Component Store	Storage Management Module
2. Submit report to Administration	Storage Management Module; Alerting Service

Hardware Test and Maintenance Window

Steps	Abstract Services
1. Activate Hot Spare, if available	Storage Management Module
2. Take down the server in question	Repository Stasis
3. Perform test suite and regular maintenance operations	System Administration Protocol
4. Report test results to Administration	Alerting Service
5. If no immediate problems are discovered, restore server to active service (or Hot Spare status)	Storage Management Module
6. Report test failure to Administration and take corrective action prescribed by Administration	PDI Module; Alerting Service
7. Document test results and maintenance activities in repository history metadata	PDI Module; Repository History

Security Audit

Steps	Abstract Services
1. Hire an independent auditor to review security procedures and protocols, adherence to procedures, physical security, and other security practices	Security Protocol; Security Audit
2. Report security audit results to Administration	Security Protocol; Security Audit
3. Document security audit results in repository history metadata	Security Protocol; PDI Module; Repository History

Irregular Events Types

Digital Object Accession

Steps	Abstract Services
1. Generate Storage Identifier(s)	Persistent Identifier Manager
2. Place records component(s) in Records Component Storage	Storage Management Module
3. Document Storage Identifier(s) in AIP	Persistent Identifier Manager
4. Add "Object Accession" event to PDI history	PDI Module
5. Place AIP and PDI in Administrative Metadata Storage	Storage Management Module
6. Update Data Management Database	Data Management Database
7. Schedule Events based on accession date	Alerting Service

Retrieval Request

Steps	Abstract Services
1. Look up the Storage Identifier for the record component	Request Service Manager
2. Retrieve record component from Records Component Storage	Request Service Manager
3. Provide record component to requesting archive function	Request Service Manager
4. Update retrieval statistics	Request Service Manager

Query Request

Steps	Abstract Services
1. Perform requested query	Search Service
2. Filter results set as necessary according to initiator's permissions	Search Service
3. Provide results set to requesting archive function	Search Service
4. Update query statistics	Search Service

Metadata Update Request

Steps	Abstract Services
1. Add a new metadata bitstream or version existing bitstream as appropriate	Storage Management Module
2. Update AIP with any new Storage Identifiers and fixity information	PDI Module
3. Update PDI with "Metadata Update Event"	PDI Module
4. Update Data Management Database	Data Management Database

Format Transform Request

Steps	Abstract Services
1. Identify all records components or metadata bistreams that are represented in the affected format	Persistent Identifier Manager, Alerting Service
2. For each bitstream, retrieve the bitstream and perform the transformation	Format Transformation Service
3. Validate the file formats of the transformation outputs; report any validation failures to Administration and take corrective action prescribed by Administration	Format Validation Service
4. Generate a Storage Identifier for the newly created bitstream	Persistent Identifier Manager
5. Store the bitstream in the Records Component Store	Storage Management Module

6. Update AIP to include the new bitstream	AIP Module
7. Update PDI with a "Format Transformation" event	PDI Module
8. Update Data Management Database	Data Management Database

Remove Record Component Request

Steps	Abstract Services
1. Confidentially destroy the records component	Retention and Disposition Module, Storage Management Module
2. Update AIP to remove the record component	AIP Module
3. Update PDI to add "Record Component Removed" event	PDI Module
4. Update Data Management Database	Data Management Database

Preservation Application Hardware Environment Replacement

Steps	Abstract Services
1. Continue to maintain Hardware Environment	System Administration Protocol
2. Acquire the new Hardware Environment	System Administration Protocol
3. Set up the new Hardware Environment with a new installation of the Preservation Application and all needed system and utility software	System Administration Protocol
4. Set up new Administrative Metadata Store and Data Management Database (and Records Components Store, if necessary)	System Administration Protocol
5. Place the Preservation Application in stasis on the old Hardware Environment	Repository Stasis
6. Update all PDI to include a "Begin Hardware Environment Replacement" event	PDI Module
7. Copy all AIPs and PDI from the old Administrative Metadata Store to the new Administrative Metadata Store. Also copy Records Components Store if necessary	Storage Management Module
8. Build the new Data Management Database from the new Administrative Metadata Store	Data Management Database
9. Test a representative sample of AIPs on the new system to ensure full functionality	AIP Module
10. After passing tests, update PDI on new server with "End Hardware Environment Replacement" event	PDI Module
11. Update Repository History record with "Hardware Environment Replacement" event	PDI Module
12. Perform full backup image of the new Administrative Metadata Store	Data Backup Protocol
13. Remove Preservation Application from stasis on new Hardware Environment	Repository Stasis
14. Re-designate new Hardware Environment as active	System Administration Protocol
15. Confidentially destroy data on old Hardware Environment	System Administration Protocol

New AIP Format and/or New Preservation Application

Steps	Abstract Services
1. Continue to maintain the old Hardware Environment	System Administration Protocol
2. Acquire a new Hardware Environment	System Administration Protocol
3. Set up the new Hardware Environment with a new installation of the new Preservation Application and all needed system and utility software	System Administration Protocol
4. Set up new Administrative Metadata Store and	System Administration Protocol

Data Management database (and Records Component Store, if necessary)	
5. Place Preservation Application in stasis on old Hardware Environment	Repository Stasis
6. Update all PDI to include "Begin Preservation Application Change" event	PDI Module
7. Transform all AIPs and PDI and submit them to the new repository. Also copy Records Component Store if necessary	Storage Management Module
8. Build the new Data Management database from the new Administrative Metadata Store	Data Management Database
9. Test a representative sample of AIPs on the new system to ensure full functionality; report any test failures to Administration and take corrective action prescribed by Administration	AIP Module; Alerting Service
10. After passing tests, update PDI on new Hardware Environment with "End Repository Application Change" event	PDI Module
11. Update Repository History with "Change Repository Application" event	PDI Module
12. Perform full backup image of the new Administrative Metadata store	Data Backup Protocol
13. Remove Preservation Application from stasis on new Hardware Environment	Repository Stasis
14. Re-designate new Hardware Environment as active	System Administration Protocol
15. Confidentially destroy data on old Hardware Environment	System Administration Protocol

New Records Component Store

Steps	Abstract Services
1. Acquire the new Storage Hardware Environment	System Administration Protocol
2. Set up the new Storage Hardware Environment	System Administration Protocol
3. Test new Storage Hardware Environment	System Administration Protocol
4. Copy the Records Component Store when necessary; See **Preservation Application Hardware Environment Replacement**	System Administration Protocol; *Also see* ***Preservation Application Hardware Environment Replacement***

Add Additional Representation Information

Steps	Abstract Services
1. Accession the digital objects representing the new RI (see **Digital Object Accession**)	*See* ***Digital Object Accession***
2. Update any objects which require links to this RI (see **Metadata Update Request**). This list of objects is provided by Preservation Planning.	*See* ***Metadata Update Request***

Change Standard Computing Platform

Steps	Abstract Services
1. Update SCP record (see **Metadata Update Request**)	*See* ***Metadata Update Request***
2. Find all records that are no longer grounded in the Knowledge Base	Knowledge Base Module
3. Report all such records to Preservation Planning	Alerting Service

Refresh Records Component Media

Steps	Abstract Services
1. Prepare new media for use	Storage Management Module
2. Test new media for manufacturing defects	Storage Management Module
3. Copy records components onto new media	Storage Management Module
4. Perform a bit-level comparison between old and new media	Storage Management Module
5. If bit-level test succeeds, redesignate new media as active storage for affected records components	Storage Management Module
6. Update PDI for all affected records with "Media Refresh" event	PDI Module
7. Document when media becomes active	Storage Management Module
8. Confidentially destroy data on old media	Storage Management Module
9. Discard or recycle old media	Storage Management Module

Respond to Checksum Failure

Steps	Abstract Services
1. Mark the record as containing compromised data	PDI Module
2. Alert Administration of this event	Alerting Service
3. Search alternate storage locations to find the record component backups	Data Backup Protocol
4. If an intact, independently stored record component is discovered, then go to **Verify Records Components**, verify component checksum.	Data Recovery Protocol; *See also* ***Verify Records Components***
5. If component checksum does not match, proceed to next appropriate independently stored record component.	Data Recovery Protocol
6. If no intact datastream is found, go to **Respond to Data Loss: Record Component Store**	Data Recovery Protocol; *See also* ***Respond to Data Loss: Record Component Store***
7. If component checksum matches, then replace current record component with alternately stored component in the active Record Component Store, and document that the Archive has repaired the record in its PDI	Data Recovery Protocol
8. Update record PDI "Record Component Repaired" event or "Record Component Corrupted" event	PDI Module

Respond to Media Failure: Record Component Store

Steps	Abstract Services
1. Mark all records with affected components as having corrupted data	PDI Module
2. Look for alternate storage locations for the data stream (such as backups)	Data Backup Protocol
3. Prepare and test new media	Storage Management Module
4. Restore onto new media all found datastreams that positively match their existing integrity information	Storage Management Module
5. Document that the Archive has repaired the records in their PDI	PDI Module
6. If any record components could not be repaired, report to Preservation Planning and go to **Respond to Data Loss: Record Component Store**	Alerting Service; *Also see* ***Respond to Data Loss: Record Component Store***

Respond to Data Loss: Record Component Store

Steps	Abstract Services
1. Notify Archive Administration and Preservation Planning of the data loss	Alerting Service
2. Document data loss in repository history metadata	Repository History
3. Document data loss in the PDI of the affected records	PDI Module

Respond to AIP Consistency Failure

Steps	Abstract Services
1. Place Preservation Application in stasis	Repository Stasis
2. Check all AIPs for consistency. Assume all media upon which consistency failures have occurred has failed; go to **Media Failure: Administrative Metadata Store**	Integrity Checking Protocol

Respond to Media Failure: Administrative Metadata Store

Steps	Abstract Services
1. Place Preservation Application in stasis	Repository Stasis
2. Acquire a new Preservation Application Hardware Environment	System Administration Protocol
3. Set up a new Administrative Metadata Store	System Administration Protocol
4. Set up the new Preservation Application Hardware Environment with a new instance of the Preservation Application and all needed system and utility software	System Administration Protocol
5. Copy all savable AIPs and PDI to the new Administrative Data Store	System Administration Protocol
6. Reconstruct missing AIPs from backup images	System Administration Protocol
7. Document missing or corrupted AIPs that cannot be restored from backup images	PDI Module
8. Document the media failure in the repository history metadata	Repository History
9. Report results of the reconstruction to Archive Administration, who will decide what further actions to take. If any missing or corrupted AIPs cannot be restored from backup images, Archive Administration will probably want to undertake **Respond to Data Loss: Administrative Metadata Store**	Alerting Service; *See also* ***Respond to Data Loss: Administrative Metadata Store***

Respond to Data Loss: Administrative Metadata Store

Steps	Abstract Services
1. Determine all records components "orphaned" by the lost AIP(s)	AIP Module
2. Generate a new "record fragment" AIP for each of the record components, including as much information as can be reconstructed or gathered from the records components, including at least the format type of the components	AIP Module
3. Document information about the data loss in the PDI for the new AIPs	PDI Module
4. Document the data loss in the repository history metadata	PDI Module

Respond to Unintentional Data Damage

Steps	Abstract Services
1. Place Preservation Application in stasis	Repository Stasis
2. Identify the scope and nature of the damage	Integrity Checking Protocol
3. Report to Archive Administration concerning the scope and nature of the damage; Administration will decide the appropriate corrective action	Alerting Service
4. Take corrective action prescribed by Administration; if data loss occurs, undertake **Respond to Data Loss: Administrative Metadata Store** or **Respond to Data Loss: Records Component Store**	*See **Respond to Data Loss: Administrative Metadata Store** or **Respond to Data Loss: Records Component Store***
5. After taking the Preservation Application off stasis, record damage and corrective actions in repository history metadata and the PDI of all affected records	PDI Module

Respond to Security Breach

Steps	Abstract Services
1. Take Preservation System offline; do *not* activate Hot Spares	Repository Stasis
2. Analyze all repository hardware to determine what machines have been compromised and to discover the nature and scope of the attack, and what actions the attacker took while he or she had access to the Hardware Environment	System Administration Protocol
3. Perform internal consistency check of all Administrative Metadata	Integrity Checking Protocol
4. Perform checksum verification of all records components	PDI module
5. Compare Administrative Metadata to a known-good backup image (taken before the attack occurred), and compile a list of all changes between the current image and the backup	Data Recovery Protocol
6. Report all findings to Archive Administration which determines if the attacker was: a. A Squatter (only using computing resources) b. A Vandal (intending to do indiscriminate damage) c. An attacker with motive against the records (intending to alter or destroy records in particular)	Alerting Service
7. Administration decides appropriate corrective actions	Security Protocol
8. Perform prescribed actions; if data loss occurs, undertake **Respond to Data Loss: Administrative Metadata Store** or **Respond to Data Loss: Records Component Store**	*See **Respond to Data Loss: Administrative Metadata Store** or **Respond to Data Loss: Records Component Store***
9. Document security breach in repository history metadata and PDI of all records	PDI Module; Repository History