2019-02 Fedora Design Summary

Design Principles

1. Minimize change to the user via the API
2. Retain URLs of migrated Fedora resources
3. Compliance with OCFL
4. Do not allow OCFL-isms from bleeding into Fedora API
5. Rebuildability
6. Performance
7. Reduce complexity of implementation

Issues being addressed

Based on feedback from users of Fedora 4 and 5, the design for the next major release of Fedora will address the following issues:

Preservation persistence

The notions of "completeness" and "transparency" are important when it comes to how a preservation repository persists its resources (metadata and binaries) to storage. The resources in storage should be "complete" in the sense that Fedora should be able to rebuild its indexes based on what is stored on disk as files. The documented transparency of those persisted files also allows for other applications to consume those resources. See section below: *O CFL Persistence*.

Query service

The ability to query Fedora for basic information regarding the contents of the repository has been a missing feature in Fedora 4 and 5. This design will include a simple query service for inspecting all of the Fedora resources or resources based on specific attributes. See section below: *Query service*.

OCFL persistence

Architecture

1. Retaining HTTP layer of existing Fedora codebase
2. Replacing ModeShape persistence with OCFL storage
3. Support for three interaction models:
    a. atomistic (implicit) - every LDP resource maps to an individual OCFL Object
    b. archive group - hierarchy of LDP resources map into a compound OCFL Object
    c. archival-part (implicit) - an LDP resource that is a constituent part of a compound OCFL Object
4. Eliminate "single-subject-restriction", i.e. support arbitrary RDF
5. Fedora-specific information to be stored in the OCFL Object in a ".fcrepo/" directory
    a. i.e. Which file is the description of another file
    b. i.e. Which file is an ACL
6. Optimizing reads/lookups with an internal database
    a. proposed database model: https://docs.google.com/document/d/1MsMfhae3thmNdoFtnTUnII3mr_-OkllRs9PvgnY1fDY/edit
7. Support for both OCFL storage hierarchies:
    a. created by Fedora

    b. created by another application (pre-existing)

Mapping between LDP and OCFL

Opt-in model

1. Fedora resources may be created with an optional "archive group" interaction model provided via headers
2. New resources created via POST or PUT to the archive group will be LDP contained by the archive group and will be stored within the OCFL object representing that archive
3. If a resource is created without the "archive" model, new resources created via POST or PUT will be LDP contained by the parent resource, but will be stored as separate OCFL objects
4. Note: user establishes interaction model at creation time. Changing the model would require additional migration tooling.

Bulk ingest

1. Faster ingest rates can be achieved by users writing OCFL-compliant content directly to disk
    a. Would require Fedora to (re)scan OCFL storage hierarchy
2. Optionally, user could write into OCFL-compliant storage in a way that includes Fedora optimizations (e.g. ".fcrepo/" directory)

Open questions

1. Role of OCFL storage roots
    a. Could be valuable for multi-tenancy, but client interaction model has not been detailed
2. What is the mapping / algorithm / relationship between:
    a. Fedora URL of LDP resource
    b. OCFL Object.ID
    c. OCFL storage path for associated OCFL Object

Implementation notes

1. Provide new implementation of fcrepo-kernel-api that interacts with OCFL persistence
2. Interactions with OCFL persistence should initially take advantage of the JHU OCFL client
3. For pre-existing OCFL storage hierarchies, Fedora-imposes the following constraint:
    a. The OCFL storage hierarchy must have a single, consistent "ocfl_layout" (i.e. the storage path mapping algorithm must be determinant)
4. Many members: performance should improve significantly since list of members will be supplied by a database index (which should support a degree of in-memory caching)
5. Deleting tombstone of OCFL Object purges the Object
6. Deleting tombstone of "constituent part" is not supported (405)

Prototyping proposal

1. Expose JHU OCFL client functionality with minimal HTTP endpoints
    a. Such an endpoint should implement minimal LDP interactions
2. Use HTTP over OCFL to test:
    a. Performance bottlenecks
    b. Scale viability (e.g. NLM migration)
    c. User expectations, ergonomics

Versioning

1. Support for two versioning models:
    a. version an object on-demand (manual versioning)
    b. version an object on-change (auto-versioning)
2. Support for toggling auto-versioning on/off
3. One-to-one correspondence between OCFL versions and mementos
4. For archive groups, any new version of the OCFL Object captures current state of the entire archive group

Versioning on-demand

1. Same as Fedora 4 and 5 version creation: POST to a resource's "/fcr:versions" endpoint to create a Memento (i.e. a new OCFL version directory)
2. Actively edited objects captured in a "cache/" directory at the sibling-level with OCFL version directories

Versioning on-change

1. Every update to a Fedora resource results in a new OCFL version directory
2. Potential downsides:
    a. Potential storage impact
    b. Potentially creates "noisy" version history

    c. Note: Transactions could mitigate "noisy" version history by grouping multiple updates in a single commit

Implementation notes

1. Same code logic used for creation of OCFL versions / Mementos in both on-demand and on-change models
2. LDP resources within a compound object should respond with a "Link" header pointing the the TimeMap of the Fedora "archive group" resource
3. POST on /fcr:versions of part resources returns a 400 response
4. GET on /fcr:versions returns a version of the "constituent part"

Migration from lower versions of Fedora to higher

1. Design
    a. Release import/export tool for each version of Fedora (4, 5, 6)
        i. Import/Export tool for a given release is able to round-trip content for that release
    b. If necessary, transform exported serialization produced from one Fedora version to the a serialization that is expected for the import Fedora version
        i. May be able to transform F3? 4, 5 directly to F6-OCFL on-disk serialization
2. Fedora resource URLs must remain unchanged during migrations
3. Persistence model of Fedora 6 should be stable enough to eliminate the need for a content migration to Fedora 7

Fixity service

1. Requirements:
    a. Check fixity of binary resource(s) by comparing computed value with stored value
    b. Check fixity of binary resource(s) given a specific set of Fedora object rdf:types
    c. Persist results of fixity check
        i. In log file?
        ii. In database?
        iii. In Fedora?
2. Scheduled fixity service:
    a. Probably not part of the core
    b. Run as a separate service (see: Riprap)
    c. Potentially implemented as a circular queue of Fedora resources, ordered by "last fixity check" date property on Fedora resource
3. Retain "fedora:hasFixityService" triple or header
4. At the OCFL-level, interest in providing fixity over an OCFL storage hierarchy

Query service

1. Should also consider this "Query Service Specification"
2. Proposal: Query service / endpoint should support the following queries:
    a. List all resources
    b. List resources by mimetype
    c. List resources by parent
    d. List resources by mimetype, parent, and modified date (<>=)
    e. List resources where modified <> x date
3. Open questions around scope of resources to be searchable
    a. Fedora resources?
    b. Resources defined in RDF documents within the repository?
    c. Hash URIs?
4. Open questions around properties to support
    a. Server-managed triples?
    b. All properties?
5. Triplestore not necessarily required

Implementation notes

1. Index of all Fedora resources would be needed to support the query service
2. Messaging model (synchronous or asynchronous) would likely be used to populate the index
3. Full-text search would be a bonus

Transaction service

1. Proposal: no change to the Fedora API spec in 6
2. We will either:
    a. align code with the (as-yet-to-be-ratified) side-car specification
    b. leave HTTP API unchanged while introducing the possibility of auto-versioning on transaction completion
3. Potentially store updates within a transaction in a "txn/" directory at the sibling-level with OCFL version directories
4. Support actions on multiple OCFL objects within a single transaction

Raw notes

1. General VA Beach Meeting notes
2. Design summary notes
3. Migration notes
4. Object modeling notes
5. Versioning notes
6. Fixity notes
7. Bulk ingest notes
8. Query service notes