Fedora 3 to 4 Data Migration

Deprecated Information

The information in this page is now **out-of-date**. See the Fedora Project Roadmap for the latest information on Fedora development.

Unlike the functionality migration, migrating fedora 3 data to fedora 4 can be lossless (and therefore reversible). This page is meant to discuss strategies and tooling to accomplish this migration.

Considerations that shape these strategies include:

- need to **preserve** fedora 3 content, history and audit trail
- ability to leverage fedora 4 features
- need to make data **accessible and functional** in the new environment
- desire to make migration easier, **faster** and less error-prone

Ideas behind migration-utils

A framework for pluggable migration tool that is based on processing of FOXML xml. Ideally this utility code could be packaged as a command line program for unix-like operation, or used as a library within more complex tools such as a camel-based migration utility.

A FOXML-based (rather than API based approach) has the advantages of:

- foxml (when exported in the "archive" context, or persisted in the low level store) is a complete representation of the object
- foxml offers a wide range of compatibility with various versions of Fedora
- foxml migration doesn't require the fedora 3 repository software to be running
- large number of existing frameworks for efficiently processing XML

Considerations:

- migration of data that's not in the repository (like configuration, global xacml policies, etc.) will require special handling
- ability to write and use plugins (special configurations) for mapping complex metadata or fedora 3 constructs into fedora 4 must be made as easy as possible since most institutions will need to write their own or adapt existing ones

Migration Model

There are several ways to expose or abstract fedora 3 objects to a migration utility and it may be that some are better suited for the types of migration necessary. For example, sequentially accessing content from within a FOXML file may prove to be the least memory-intensive and fastest way to process content, but if information in a datastream (like RELS-EXT) is needed to inform decisions about how to handle earlier content, such benefits may be negated in common use cases. Furthermore, beyond simply processing objects sequentially, it may prove beneficial to process objects in a certain order, or to provide random-access to other resources within the fedora 3 repository being migrated.

model	description	considerations	status
sequential access	Streaming the FOXML in the order present on disk to the routines or code that does the migration.	This model minimizes memory usage and likely maximizes processing speed. It can also more easily force higher-level routines to acknowledge all content, preventing inadvertent lossyness. Complex higher-level routines would have to accept the data in the serialization order and couldn't easily alter processing based on content serialized deeper in the foxml datastream.	A proof-of-concept implementation has been written.
whole object access	Expose access to the whole object in some form. Pieces (like datastream content) would likely have to be accessed as needed from disk.	This model allows for more flexibility in higher level migration routines as content could be accessed in whatever order at the expense of a higher memory footprint and less coercion to consider every element within the original fedora 3 object.	An implementation of this is being built on top of the sequential access model.
versioned object abstraction	Fedora 3 allows versioning of datastreams but not object properties, though it does maintain a simple audit record of those object property updates. In migration scenarios where metadata from multiple sources (different datastreams or object properties) are to be represented in the Fedora 4 properties, it may make sense to have an abstraction representing a version of the fedora 3 object (and all its datastreams) in time.	This model would take care of a huge amount of the heavy lifting associated with version migration and the likely case where the DC datastream, RELS-EXT datastream and object properties would all find themselves represented as fedora 4 properties.	An implementation of this is being built on top of the whole object model.

Implementation

GitHub