Functional Requirements for the Migration to Fedora 3.0

The 5.0 release of the RUcore repository software will not include new functionality, but will include major changes to the infrastructure of the repository. There are two main goals in this release: 1) to migrate the repository objects with current functionality from Fedora 2.1 to Fedora 3.0, and 2) to prepare the infrastructure to support the enhanced functionality expected in 5.1 and later releases.

1. Migration to Fedora 3.0

1.1 Content Model Architecture

Fedora's 3.x release itself involves some major changes to its current infrastructure. Chief among these is the use of the so called Content Model Architecture (CMA), the most immediate affect of which is to replace the disseminator architecture used in earlier releases of Fedora. Repository operations and behaviors are now attached to objects at the content model level. Where earlier versions of Fedora recognized three types of object: ordinary data objects, behavioral definition objects (BDef), and behavior mechanism objects (BMech), Fedora 3 recognizes a fourth type, content model (CModel) objects.[1] The relations of these different types of object are now to be expressed as RDF statements in RELS-EXT Datastreams (e.g. "fedora-model:hasModel" for data objects, "fedora-model:hasSDef" for CModel objects, and "fedora-model:isContractor" for SDep or Service Deployment objects).[2] All objects will now be associated with a content model, though as with the "default disseminator" of earlier versions, there could be a transparent default content model for simple objects that do not require special dissemination views. Most of the objects in the RUcore repository do not require special dissemination, and thus would be able to use such a default content model. Fedora 3 provides a migration utility that should help to create content model objects replicating

[1] In beta releases the nomenclature of BDef and BMech for these objects was supported, but as of the 3.0 release in late July these are known as "Service Definition" (SDef) and "Service Deployment Mechanism" (SDep) objects.

[2] The RELS-EXT datastream assigning a content model looks something like this:
```
<foxml:datastream ID="RELS-EXT" STATE="A" CONTROL_GROUP="X" VERSIONABLE="true">
<foxml:datastreamVersion ID="RELS-EXT.0" LABEL="Relationships" CREATED="2008-05-
05T20:10:21.901Z" MIMETYPE="text/xml" SIZE="236">
<foxml:contentDigest TYPE="DISABLED" DIGEST="none"/><foxml:xmlContent>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:fedora-
model="info:fedora/fedora-system:def/model#">
<rdf:Description rdf:about="info:fedora/rutgers-lib:XXXYYYZZZ">
<fedora-model:hasContentModel rdf:resource="info:fedora/rutgers-lib:Book-cmodel"/>
</rdf:Description>
</rdf:RDF></foxml:xmlContent></foxml:datastreamVersion></foxml:datastream>
```
This code associates an as yet unnumbered object with a generic book content model Rutgers-lib:201856. The string "XXXYYYZZZ" is replaced with the Fedora PID of a new object upon ingest. Markup like this would have to be inserted in pre-ingest objects by the WMS, much in the way it currently inserts code identifying disseminators. The WMS would reserve a Fedora PID and use that in place of the XXXYYYZZZ string in pre-ingest objects. Note that this datastream, unlike disseminator datastreams, is *versionable* so that the content model and its attendant dissemination could easily be changed if need be.

the functionality of the relatively few disseminators currently in use by RUcore. This procedure is still being tested.

The new content models are not simply replacements for disseminators however. The Fedora Content Model Architecture web page lists two combinable "meanings" for the term Content Model:

1) Content structure as used by publishers and other traditional content-related professions
2) A computer model describing an information representation and processing architecture

The second definition closely approximates the "disseminator" function of the CMA that has been discussed above, while the first definition could suggest something more like the term "object architecture" as used in the context of the RUcore repository. Our early experiments with the migration and automated creation of CModels for RUcore objects, however, suggest that the migration scripts, which create CModels based on the differing sets of XML datastreams present in existing objects[3], are not capable of creating the sort of "semantic" CModels that would accord with our notions of object architecture. A recent test of the automated migration script on lefty64 based on about 3,100 objects carried over from lefty generated almost 400 distinct CModels. We need to design our own generic CModels and assign them to objects meeting certain sets of criteria. We are currently testing the implementation of such an approach, and it has had promising results so far, but it will almost certainly complicate the migration process beyond the use of the migration script, perhaps requiring an additional filter of the RELS-EXT Datastreams applied at some point in the migration. As we continue to assess the functionality of the content models, we envision having all objects, rather than simply those requiring dissemination, assigned content models based on our current definitions of "object architectures".[4] As this approach would have an impact on the WMS in addition to the migration of current objects, we will need to affirm that some kind of useful validation capability will eventually be gained in the process.

1.2 Migrating objects and ingest targets to FOXML 1.1 and METSFedoraExt 1.1

Fedora 3 makes use of updated XML schemas for FOXML as well as the Fedora version of METS now known as METSFedoraExt.[5] This will have consequences for us not only during the migration process, but also going forward with the WMS and dlr/EDIT Fedora management system. The migration scripts allow objects in an earlier Fedora repository to be upgraded "in place", a process that changes the XML encoding of the objects while leaving them in their current directories. The changes to the schema are relatively subtle, but will need to be accounted for in our XML-creation programs. The foxml:digitalObject element, for instance, gets a new, required "VERSION" attribute as

[3] For example, objects with three digiprov sections generate a different content model from essentially similar objects that happen to have two or four digiprov sections.

[4] The WMS will begin using the element <rulib:contentModel> in place of the current <rulib:objectArchitecture> in technical metadata sections.

[5] While export in the new METS format still works as expected, we have experienced problems ingesting METSFedoraExt-1.1 into the Fedora 3.0 beta repository. We do not know if this is an issue that will be corrected in the production version.

well as a new schema location. The foxml:disseminator element is no longer accepted and will need to be replaced by a RELS-EXT datastream referencing a particular CModel object. The foxml:datastreamVersion element gets a new, empty element, foxml:contentDigest, designed to contain attributes with signature information, but which is marked "DISABLED" by default for compatability with older objects.[6] Managed datastreams are not affected by the migration script and remain in their current directories.

1.3 Adjusting to and using the new Fedora API

Once again, the new version of Fedora includes changes to the API used to access or manage various web services. Many of these changes, such as the addition of the checksumType and checksum parameters to various datastream methods, were already in place for the 2.2 version of Fedora, but are nonetheless new to us as we never implemented that version. There is a new start date parameter for the purgeDatastream method, and in Fedora 3 a new ownerID parameter for the modifyObject method.[7] Fedora 3 also includes a new set of "relationship methods" (addRelationship, getRelationships, and purgeRelationship) that we might use to manage or extend RELS-EXT sections that designate a content model or membership in a group of related objects.[8] Fedora 3 contains an enhanced REST API including a number of API-M methods previously only available through SOAP. Many of these appear not to have been implemented yet, and while the others are worth investigating for possible use in the future, we will probably not be using them for the initial migration. There are plans to include such interesting features as a Java Messaging Service (JMS) and validation testing of things like conformance to content models, but there is no documentation at the present time to suggest how these features might be expected to work.

1.4 Concomitant software infrastructure upgrades

The 5.0 RUcore release will involve other software infrastructure upgrades in addition to the migration to Fedora 3. Chief among these will be the move to PHP 5 and the use of a 64 bit architecture for development. PHP 5 migration will involve testing and in some cases altering user applications in the WMS, the dlr/EDIT Fedora management system, the statistics and notification system, and the public search and display portals. PHP 5 includes a new and improved SOAP module that may be of use for applications needing to interact with the Fedora repository. There is now also a Perl SOAP::Lite "stub" class for accessing the entire Fedora API. As this is derived directly from the Fedora WSDL

[6] See the "purple" code in the foxml example in an earlier footnote. Though for now we are not changing our approach to signature testing, we may still want to make use of this element at some point in the future to store the SHA1 signatures of archival datastreams in a place where Fedora's new signature checking methods can find them.
[7] For historical reasons, we have avoided purging datastreams based on date/time attributes, which were unreliable in earlier versions of Fedora. We might investigate possible uses for the new ownerID attribute, which is currently left blank in migrated test objects on lefty64. The Fedora 3 documentation notes, for instance, that this may be a string of comma separated values that could be used in XACML policies.
[8] RELS-EXT sections are created either on ingest or using the addDatastream method.

file, it gives us a flexible means of keeping up with any future changes that may be made to the Fedora APIs.

2. Infrastructure Preparation to Support Future Functionality

2.1 Implementation of XACML policies for authentication and authorization

The 5.0 RUcore release will be the first to support XACML policies in Fedora. XACML has been part of Fedora since the 2.1 release, but our earlier versions of RUcore, which supported only open access, have not used policy enforcement. This will change in the new release so that we can use XACML for fine-grained authorization at the datastream level in projects such as Rutgers ETDs as well as the new NJVid project. The ETD project requires an elegant mechanism for embargoing certain dissertations for a period of time, while NJVid envisions an elaborate authentication and authorization structure for allowing videos to be streamed to differing classes of users authenticated by partner institutions.[9] We are currently experimenting on lefty64 with the use XACML policies.

Fedora ships with a set of default repository policies that are very restrictive for the API-M and quit unrestrictive for API-A methods. Users are expected to modify these policies to suit their own requirements. Policies may also attach to individual objects in various ways: 1) through an external XML file stored with a filename related to the object's Fedora PID in an object policies directory defined in the Fedora configuration file, or 2) as a datastream of the object with the datastream ID "POLICY" using any of the four possible control groups (X, M, E, or R)[10].

The first option has several drawbacks, including the fact that such policies are loaded only when the Fedora server is started and the difficulty of maintaining such a system on a large scale and over time. The second option has the benefit of allowing policies to be assigned dynamically on ingest or through the API-M addDatastream method. This may be the preferred method for assigning policies to ETD objects that have embargoes for certain time periods based on a time related to the student's date of graduation. Thus such policies, though they may be based on templates, would have to be tailored to particular objects at the time of ingest. In most other circumstances, however, we envision a limited number of policies each being used for large sets of objects. In such a circumstance, use of the X or M control groups would necessitate awkward repetition of code in the workflow management system or at some later stage for individual objects and thus difficulty of maintenance. The E or R control groups would allow individual policy datastreams to be stored in an editable form either outside the repository or within

[9] At present we envision using Shibboleth to provide authentication across institutions, though it is not yet clear how it will be implemented. Fedora's XACML functions will be used to handle the authorization of access to datastreams, and will be implemented in a way that is agnostic about the ultimate means of authentication.

[10] X stands for XML code that is part of the object file itself, such as the DC or MODS datastreams. M stands for an external file of some sort associated with the object and managed internally by Fedora, such as our presentation datastreams or METS structure map files. E stands for a similar external file accessible through the http protocol and left outside the repository for storage though retrieved and managed transparently by Fedora upon access. R stands for an external file accessible through the http protocol to which Fedora redirects users upon request.

an easily manageable set of special policy objects in the RUcore or another Fedora repository.

Fedora ships with a repository policy that does not allow datastreams with the ID "POLICY" to be accessed. If we decide to keep this policy, any policies referenced in E or R datastreams would have to have an ID other than "POLICY" (say, "POLICY1") in the context of their own policy objects, though they would be referenced by other objects with ID "POLICY". This will be good practice whether or not we decide to keep the default policy viewing restrictions. Our anticipated use of individual object policies in ETD objects would seem to require that such policies be visible for inspection and editable through one of the current editing interfaces, but this can be accomplished without changing the current restrictions on API-A access to POLICY datastreams.

2.2 Exploration of alternative object architectures

In preparation for the management of many large video objects and other "unusual" objects such as electronic journal articles or EAD finding aids, we need to explore alternatives to the fairly rigid system of "M" or managed datastreams that that we have used until now. We have already begun to discuss using "R" (redirected) datastreams for large archival video files, and we need to start modeling such objects on the test server, as well as experiments with "R" datastreams for electronic journal articles and EAD files where maintaining external link relationships for presentation datastreams can be important. We should also begin experimenting with a means of federating searches across distributed Fedora installations. These tests will have implications for various systems, such as the current checksum testing system and the workflow management system.

2.3 Storage architecture

The following are major assumptions regarding the storage architecture and the 5.0 RUcore release: We will need to treat archival masters for large files using the Fedora "redirect" mode. In particular, the NJVid release will need this capability for video archival masters. Note that WMS has already been updated to allow any archival master to be external. The WMS will have to mark such datastreams as "R" and create them with IDs in the form "RARCH1" rather than "ARCH1" to identify them more readily as "external" files to management programs such as the signature verifyer.

For the 5.0 RUcore release, the "external" file system will be local to the Fedora server. In addition to changes for WMS, the signature verifying code will need to find the archival master as a managed datastream or an external datastream.[11]

[11] We have experimented with Fedora's own checksum mechanism but found it not suitable for our uses at the present time. We tried putting checksums in the foxml:contentDigest element for certain datastreams. Such checksums are tested on ingest and may be tested by Fedora at any time using the compareDatastreamChecksum method of API-M in a way that is attractively agnostic about server location. Thus E and R datastreams with checksums are tested along with M or X datastreams in a way that is transparent to the user. For E and R datastreams that reside in another Fedora instance, the checksum testing is actually off-loaded onto the server in question. For R datastreams that do not reside in a Fedora instance, the datastream is first pulled onto the accessing Fedora server and the testing is done on a copy there. This approach, however, has the disadvantage of unacceptably slowing the ingest times of such

For external archival masters (RARCHs), the archival master file along with presentation files and the checksum will need to be located in the external file system. WMS will upload the presentation files, point to the archival master, and include the checksum in technical metadata. This process requires that these files and the checksum be pre-loaded in the external file system.

The question of whether we need a backend storage server will need further discussion. In particular, we need to be part of the discussion with NJVid as to what storage system they will purchase.

3. Issues Going Forward

We are still in an early testing phase using the first public release version Fedora 3. After an initial period of difficulties, we have had success modeling the new API methods for dlr/EDIT, so that the functionality on the 3.0 test machine now replicates that of the 2.1 production server.[12]

The old disseminators are now functioning on the test machine using the new content model architecture, and we have successfully modified these models and added them to existing objects to provide dissemination. The migration utility has produced too many content models replicating the same functionality for disseminators and unnecessarily distinguishing objects with similar semantic architectures, but we have discovered a means of "seeding" that utility with archetypical objects that promises to mitigate this problem. We need to explore the management of the content models more thoroughly, and to see if it is feasible to validate somehow our "object architecture" models.

Our early tests with XACML show some promise, but we need to experiment with implementing role-based policies in an authentication-neutral setting, which could later be hooked up to an authentication system (we expect this to be Shibboleth) yet to be built or not yet ready for testing. Since a number of our current and anticipated user interface functions (e.g., streaming video) access datastreams outside of the usual Fedora services, we need to build an efficient system of using API methods to probe Fedora for policy restriction information within the user interface.

3.1 Questions and resolutions going forward

These are some questions and resolutions grouped by category. At the end of each requirement there will be an indicator of whether it will be implemented in the 5.0 or 5.1.

CMA

1) We will point to a content model on ingest by adding the special RELS-EXT elements, such as the one in the early footnote, to our objects. (5.0)

objects, and as it still relies on http sockets, it cannot deal with files larger than a certain size.

[12] Ingesting METS objects is still an unresolved problem, but this may be simply an issue with the beta version of Fedora 3.0. Since there is no mention of METS being deprecated in the current documentation, we look for it to be fixed in the production version.

2) All objects will require a content model. We will be adding appropriate “object architecture” content models on ingest. (5.0)
3) We should create content models that represent our object architectures. It will require experimentation with content modeling as well as analysis of the essential characteristics of the current object architectures to be used in the modeling. Content models chiefly describe certain expected datastreams and their mime-types, but we have not yet discovered a means of enforcing validation.[13] We assume that Fedora will provide a means of validating content models in its final 3.0 release, and will go forward on that assumption. We have tested this using various methods of assigning content models to objects. RELS-EXT elements can be included on ingest, and after ingest we can also use the addDatastream method to add content model relationships to already ingested objects. (5.0)
4) The old disseminators will work in Fedora 3.0 after migration through their related content models, though we have to call them somewhat differently since the disseminator PIDs no longer reside immediately in the object’s XML. (5.0)

WMS Impact

1) The WMS will have to replace its current disseminator datastreams with RELS-EXT elements pointing to content models controlling the equivalent disseminations.[14] Such content models do not currently affect ingest from the point of view of the WMS.[15] In fact, it is still possible to ingest objects that reference non-existing content models. We need to explore this functionality more fully. (5.0)

---

[13] It would appear from our experiments that functioning content models (i.e., models that drive disseminations) can be assigned currently to objects whose datastreams do not reflect the datastreams identified in the content models themselves.

[14] The following is an example of a new RELS-EXT datastream assigning a collection-disseminating content model (the 123456789 identifier would actually be a Fedora PID number reserved in advance by the WMS):

```
<foxml:datastream ID="RELS-EXT" STATE="A" CONTROL_GROUP="X" VERSIONABLE="true">
<foxml:datastreamVersion ID="RELS-EXT.0" LABEL="Relationships" CREATED="2008-05-
05T20:10:21.901Z" MIMETYPE="text/xml" SIZE="236">
<foxml:contentDigest TYPE="DISABLED" DIGEST="none"/><foxml:xmlContent>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:fedora-
model="info:fedora/fedora-system:def/model#">
<rdf:Description rdf:about="info:fedora/rutgers-lib:123456789">
<fedora-model:hasModel rdf:resource="info:fedora/rutgers-lib:Collection-cmodel"/>
</rdf:Description>
</rdf:RDF></foxml:xmlContent></foxml:datastreamVersion></foxml:datastream>
```

This replaces the following lines used for the disseminators of current collection objects:

```
<foxml:disseminator ID="DISS1" BDEF_CONTRACT_PID="rutgers-lib:1565" STATE="A" VERSIONABLE="true">
<foxml:disseminatorVersion ID="DISS1.0" LABEL="DLR Dynamic Collection" B
MECH_SERVICE_PID="rutgers-lib:1567" CREATED="2003-06-05T06:32:00.000Z">
 <foxml:serviceInputMap>
 <foxml:datastreamBinding KEY="OBJSTRING" DATASTREAM_ID="SMAP1" LABEL="Binding for DR Collection"
ORDER="0"/>
 </foxml:serviceInputMap>
 </foxml:disseminatorVersion>
 </foxml:disseminator>
```

[15] The rdf:about attribute (green code in the foxml example footnotes), however, must properly reference the PID that is about to be created for the object, and thus the WMS must probe for this information in advance using the getNextPID method. The dlr/EDIT ingest script will also do this for objects with the XXXYYYZZZ place-holding strings.

2) The WMS should begin planning to be able to point to external XACML policy objects as datastreams and should also develop the capability to include individual policies with objects such as the ETD objects requiring specific embargoes. XACML policy objects will already have been created and ingested separately, and it will also be possible to add datastreams referencing these objects to individual objects or collections of objects in the Fedora repository. The ETD policies can be based on templates filled in by the ETD application and delivered either as inline XML or managed XML files in the manner of the current SMAP1 files. (5.1)
3) The WMS should point to external archival master datastreams with IDs in the form RARCH1, RARCH2, etc. This will be a requirement at least for video objects in the next release, and we will need to build in the necessary flexibility in handling control groups. This will also make it easier to distinguish external archival files in the checksum testing program. (5.0)
4) As noted above, the WMS will begin using the element <rulib:contentModel> in place of the current <rulib:objectArchitecture> in technical metadata sections. Current <rulib:objectArchitecture> will have to be migrated as part of the filtering process. (5.0)

Migration

1) We have determined that objects in a Fedora 2.1 instantiation can be migrated in place to 3.0, but we will still require a full re-ingest to allow object datastream IDs to be regularized in preparation for the Content Models. Any additional filtering or work on the content models will have to be done before re-ingestion into Fedora 3.0. The objects in the repository will be pulled to the SCC, filtered, migrated to 3.0 and then exported for re-ingestion into the Fedora 3.0 instance on mss3. (5.0)
2) Any software that uses the Fedora API (access or management) will require changes to accommodate Fedora 3.0. The software calling on disseminators will require a new syntax and a new means of acquiring information about a particular disseminator, as this information is not longer stored in the object. Software accessing individual datastreams will require an efficient means of polling and interfacing with any XACML policies for those datastreams so that user links can be created consistent with the requirements of such policies. (5.0)
3) WMS editing functions for Fedora objects will continue to use Javabridge in the 5.0 release. It will need to revise its SOAP syntax to accommodate the new Fedora API. We will need to move from the 3.x version of Javabridge to the 5.2.2 version that has been designed to work with PHP 5. (5.0)

(JAT - August 20, 2008)