Migrating from Fedora 3 to 4

Now With More Hydra
Goals for the Session

Understand the basic conceptual models underlying Fedora 3/CMA, Fedora 4, and PCDM

Work through a rudimentary migration exercise with Hydra/Fedora-Migrate

Explore possibilities for enhancing data in Fedora 4
Differences Between Fedora 3 and 4
Conceptual Models of Repository Resources

Fedora 3
- Content Model Architecture
- Objects: Collect bytestreams & properties
- Datastreams: Bytestreams in context of an object, with some properties

Fedora 4
- Linked Data Platform
- LDP RDF resources (objects & containers)
- LDP non-RDF binaries (& description)
What About **PCDM**?
Organization of Repository Entities

Fedora 3: Flat
- Objects and datastreams at the top level
- No inherent tree structure

Fedora 4: Hierarchy Possible
- Containers and binaries in a hierarchy
- All resources descend from a root resource
That’s not really even organization

Right, in PCDM we have ORE proxies

“There’s really no hierarchy in a bucket.” ~ Andrew Woods
“What if you put a bucket in your bucket?” ~ Ben Armintor
Storage of Repository Data

Fedora 3: Akubra
- Objects directory and datastreams directory
- Both objects and datastreams are in a PairTree

Fedora 4: Infinispan & other MODEism
- Containers in a database (e.g. LevelDB)
- Datastreams in a PairTree directory
Identification of Repository Resources

Fedora 3: PID
- Objects have Persistent Identifiers (PID)'
- Uniform structure
- An object’s PID can never be altered

Fedora 4: Path
- Resources have a repository path
- This can be user-defined or generated via an ID-minter
## How Do These Concepts Correlate?

<table>
<thead>
<tr>
<th>Fedora 3/CMA</th>
<th>Fedora 4/LDP</th>
<th>PCDM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>RDFSource/Container</td>
<td>AdminSet/Collection/Object</td>
</tr>
<tr>
<td>Datastream</td>
<td>NonRDFSourced</td>
<td>File</td>
</tr>
<tr>
<td>PID</td>
<td>Path</td>
<td>“id”</td>
</tr>
<tr>
<td>Akubra (local)</td>
<td>Infinispan (clusterable)</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Data Mapping
## Mapping Properties - Objects

<table>
<thead>
<tr>
<th></th>
<th>Fedora 3</th>
<th>Fedora 4</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PID</strong></td>
<td>PID</td>
<td>dc:identifier</td>
<td>prefix:1234</td>
</tr>
<tr>
<td><strong>State</strong></td>
<td>state</td>
<td>fedora-model:state</td>
<td>fedora-model:Active</td>
</tr>
<tr>
<td><strong>Label</strong></td>
<td>label</td>
<td>dc:title</td>
<td>Some Title</td>
</tr>
<tr>
<td><strong>Created Date</strong></td>
<td>createdDate</td>
<td>fedora:created</td>
<td>2014-01-20T04:34:26.331Z</td>
</tr>
<tr>
<td><strong>Modified Date</strong></td>
<td>lastModifiedDate</td>
<td>fedora:lastModified</td>
<td>2014-01-20T04:34:26.331Z</td>
</tr>
<tr>
<td><strong>Owner</strong></td>
<td>ownerID</td>
<td>fedora:createdBy</td>
<td>Chuck Norris</td>
</tr>
</tbody>
</table>
Mapping Properties - Datastreams

<table>
<thead>
<tr>
<th></th>
<th>Fedora 3</th>
<th>Fedora 4</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSID</td>
<td>ID</td>
<td>dc:identifier</td>
<td>prefix:1234</td>
</tr>
<tr>
<td>State</td>
<td>state</td>
<td>fedora-model:state</td>
<td>fedora-model:Active</td>
</tr>
<tr>
<td>Versionable</td>
<td>VERSIONABLE</td>
<td>fedora:hasVersions</td>
<td>true</td>
</tr>
<tr>
<td>Label</td>
<td>label</td>
<td>ebucore:filename</td>
<td>Some Title</td>
</tr>
<tr>
<td>Created Date</td>
<td>createdDate</td>
<td>fedora:created</td>
<td>2014-01-20T04:34:26.331Z</td>
</tr>
<tr>
<td>Modified Date</td>
<td>N/A</td>
<td>fedora:lastModified</td>
<td>2014-01-20T04:34:26.331Z</td>
</tr>
<tr>
<td>Mimetype</td>
<td>MIMETYPE</td>
<td>ebucore:hasMimeType</td>
<td>image/jpg</td>
</tr>
<tr>
<td>Size</td>
<td>SIZE</td>
<td>premis:hasSize</td>
<td>50000</td>
</tr>
</tbody>
</table>
RDF Isn’t Entirely New to Fedora

http://localhost:8080/fedora-3.8.1/risearch

select $p $o from <#ri> where <info:fedora/archives:1419123/descMetadata> $p $o
Fedora 3 Sources of RDF Properties

Fedora Object Property Sources
• profile properties
• RELS-EXT
• DC
• CMA

Datastream Property Sources
• profile properties
• RELS-INT
• CMA
Containment and Structure in FCR 3

- Hints in the core RDFS vocabulary
- Sometimes implemented via Services
- or “Enhanced” content models in FCR 3.4+
- Frequently located in the application layer
Hydra Migration Tools

The Cleverly Named Fedora-Migrate
Learning Outcomes

- Fedora-Migrate Advantages & Disadvantages
- Learn basics of ActiveFedora 9 modeling
- Use fedora-migrate basic features
- Become familiar with fedora-migrate hooks
- Incorporate PCDM via hydra-works
Fedora-Migrate

Advantages, Disadvantages, Example Project
Fedora-Migrate: Advantages

You're soaking in it!

https://github.com/projecthydra-labs/fedora-migrate

- Built around the Rubydora library of Hydra <= 8
- Make data accessible and functional in the new environment
- Run migration on the stack that apps will be built on
- Very customizable
- Simplest use cases have convenient Rake support
Fedora-Migrate: Disadvantages

- Not built for speed
- Makes some assumptions about FCR 3 relationships that may require customization
  - Object-to-Object relations
  - Unidirectionality, not spidering
- No RELS-INT out of box
- No DC out of box
- Only file containment out of box
- Broader difficulty of PID to Path mapping
Fedora-Migrate: Example Project

- foxml source from [https://github.com/barmintor/usna_demo_hydra8](https://github.com/barmintor/usna_demo_hydra8)
  - already cloned on the vagrant
    - vagrant ssh
    - > cd fedora-migrate-workshop
    - > git pull origin # to make sure it's up to date
    - … or clone on your machine if you prefer to edit there
Here's an example rake task for migrating objects by ns:

desc "Migrate all my objects"
task migrate: :environment do
  Work.name
  GenericFile.name
  Collection.name
  AdministrativeSet.name
  # a convenient but difficult to extend migration convenience method
  usna = FedoraMigrate.migrate_repository(namespace: "usna",options:{})
  archives = FedoraMigrate.migrate_repository(namespace: "archives", options:{})
  report = FedoraMigrate::MigrationReport.new
  report.results.merge! usna.report.results
  report.results.merge! archives.report.results
  report.report_failures STDOUT
end
It will also be convenient to be able to delete and reset:

desc "Delete all the content in Fedora 4"
task clean: :environment do
  ActiveFedora::Cleaner.clean!
end

This duplicates the fedora:migrate:reset Rake task. Both of these tasks can be loaded from a file under lib/tasks with the 'rake' extension.
Fedora-Migrate: Example Project

checkpoint branch:
fedora-migrate/master

has no ActiveFedora models

edits lib/tasks/migrate.rake to include clean & migrate tasks

adds some helpful overrides to FedoraMigrate methods to the rake task file
Rudimentary ActiveFedora Modeling
Rudimentary ActiveFedora Modeling

Candidate models are identified by name

Given a CModel info:fedora/afmodel:GenericFile
Fedora-Migrate will look for a model called GenericFile
The model must inherit from ActiveFedora::Base
FCR 3/4 source indicate model in RELS-EXT fedora-model:hasModel
FCR 4 source also indicates types in primaryType and mixinTypes

Datastreams are modeled by File containment

Given a Fedora 3 object that has a datastream ‘content’
Fedora-Migrate will migrate if the Fedora 4 model contains a ‘content’ resource
Assuming the ‘content’ resource class inherits from ActiveFedora::File
Rudimentary ActiveFedora Modeling

Edit app/models/generic_file.rb

class GenericFile < ActiveFedora::Base
  contains 'content',
  autocreate: false,
  class_name: 'ActiveFedora::File'
end

Consider this very basic model, and look at the Fedora 3 fixtures. What other models do we need to represent? What files ought they contain? Try migrating the descMetadata datastream.

You should be able to run rake clean & rake migrate as you iterate.
Rudimentary ActiveFedora Modeling

In the rest of the workshop, we'll want a little more control over the migration. We'll get this flexibility by calling the Fedora::Migrate movers individually. Edit lib/tasks/migrate.rake to run the movers in an editable Proc:

```ruby
Collection.name
AdministrativeSet.name
migration = Proc.new do |pid|
  source = FedoraMigrate.source.connection.find(pid)
  target = nil # has not yet been migrated!
  options = {}
  mover = FedoraMigrate::ObjectMover.new(source, target, options: options)
  mover.migrate
  target = mover.target
  mover = FedoraMigrate::RelsExtDatastreamMover.new(source, target).migrate
end
```
And call the Proc for each of the objects in our example - Edit lib/tasks/migrate.rake:

migration = Proc.new do |pid|
  # snipping Proc body for slide
end

assets =
  ["usna:3","usna:4","usna:5","usna:6","usna:7","usna:8","usna:9"]

works =
  ["archives:1408042","archives:1419123","archives:1667751"]

collections =
  ["collection:1","collection:2"]

assets.each { |pid| migration.call(pid) }
works.each { |pid| migration.call(pid) }
collections.each { |pid| migration.call(pid) }
Rudimentary ActiveFedora Modeling

The sample data includes 4 FCR 3 CMModels:
● GenericFile
● Work
● Collection
● AdministrativeSet*

The example migrations will be smoothest if all of them are at least minimally modeled in ActiveFedora (though workshop doesn't do much with the AdministrativeSet object).
Rudimentary ActiveFedora Modeling

Checkpoint branch:
fedora-migrate-workshop/migrate-simple

includes very simple models corresponding to the sample FCR 3 CModels

these models mix-in Hydra::Works behaviors that will be used later

edits lib/tasks/migrate.rake to run movers individually
Modeling RDF Properties in FCR 3 Datastreams
Modeling RDF Properties in FCR 3 Datastream

Once you have basic models working with the migration task, try to migrate RDF data as properties rather than files by passing a : convert option to the RepositoryMigrator or the ObjectMover.

Look at the migrated objects to see where the models need to be elaborated to support new properties. Also note that DC is not migrated by default.
Modeling RDF Properties in FCR 3 Datastreams

Some of the objects have description stored in a datastream called 'descMetadata'.

We can migrate this data simply as a contained File or, because it is RDF properties, store the properties "natively" on the FCR 4 objects.
Modeling RDF Properties in FCR 3 Datastreams

The target properties must be defined on your models:

```ruby
class Work < ActiveFedora::Base
    property :identifier, predicate: ::RDF::Vocab::DC.identifier do |index|
        index.as :symbol, :facetable
    end

    property :title, predicate: ::RDF::Vocab::DC.title do |index|
        index.as :stored_searchable, :facetable
    end

    property :creator, predicate: ::RDF::Vocab::DC.creator do |index|
        index.as :symbol, :facetable
    end

    property :created, predicate: ::RDF::Vocab::DC.created do |index|
        index.as :stored_sortable, type: :date
    end
end
```
Modeling RDF Properties in FCR 3 Datastreams

Fedora-Migrate will then convert RDF properties if an option is passed for the appropriate datastream. Edit your rake task:

```ruby
source = FedoraMigrate.source.connection.find(pid)
target = nil # create a new target
options = { convert: "descMetadata" } # map DS as properties
mover = FedoraMigrate::ObjectMover.new(source, target, options)
mover.migrate
```

... then run rake clean && migrate. Make sure the options hash is passed correctly (no {options: ...} key should be used).
Modeling RDF Properties in FCR 3 Datastreams

Checkpoint branch: fedora-migrate-workshop/migrate-metadata

defines properties for all the descMetadata statements on the Work model

edits lib/tasks/migrate.rake to include the convert options
Customizing Fedora- Migrate with Hooks
Customizing Fedora-Migrate with Hooks

Hooks are defined in FedoraMigrate::Hooks

Methods similar to action filters on Rails controllers, or callbacks on ActiveRecord objects.

Mover#migrate implementations follow this pattern:
1. before hook
2. migrate action
3. after hook
4. save
Customizing Fedora-Migrate with Hooks

Define a state property on your models:

class Work < ActiveFedora::Base
  include Hydra::Works::WorkBehavior
  property :state,
    predicate: ActiveFedora::RDF::Fcrepo::Model.state,
    multiple: false do |index|
      index.as :symbol, :facetable
    end
end

You'll need to add this property to all 4 models!
Customizing Fedora-Migrate with Hooks

Modules like this represent RDF vocabularies:

class Work < ActiveFedora::Base
  include Hydra::Works::WorkBehavior
  property :state,
    predicate: ActiveFedora::RDF::Fcrepo::Model.state,
    multiple: false do |index|
      index.as :symbol, :facetable
    end
end

The URI objects for the RDF properties and instances are accessible as properties (above) or as a hash ( ::Model[:state] ).
Customizing Fedora-Migrate with Hooks

Override a hook to migrate object state:

```ruby
module FedoraMigrate::Hooks
  def after_object_migration
    states = {'A' => :Active, 'I' => :Inactive, 'D' => :Deleted }
    if states.has_key? source.state
      state = states[source.state]
      target.state = ActiveFedora::RDF::Fcrepo::Model[state]
    end
  end
end
```

```
rake clean && migrate
```
Customizing Fedora-Migrate with Hooks

Checkpoint branch: fedora-migrate-workshop/migrate-hook

defines a state property in the 4 ActiveFedora models

edits lib/tasks/migrate.rake to set the state property in an after_object_migration hook
PCDM via Hydra-Works
Hydra-Works brings an implementation of PCDM to ActiveFedora. This impacts the way that membership and structure are modeled: It introduces LDP DirectContainers for the former and Proxies for the latter.
If we were starting from scratch, we would add Hydra::Works model mixins to our models, identifying their PCDM role as appropriate.
PCDM via Hydra-Works

Collection maps to pcdm:Collection

Work and GenericFile are both types of pcdm: Object

AdministrativeSet was borrowed directly from PCDM
A `pcdm:FileSet` is a group of related Files, typically a single master File and its derivatives. These Files can be immediately contained, or be aggregated FileSets. Our corresponding model is `GenericFile`.

A `pcdm:Work` is intended to represent "intellectual entities" or "objects". Its members may be FileSets or other Works. This corresponds to our Work model.
Hydra::Works::FileSetBehavior
- adds directly contained Files via properties "original_file", "thumbnail" and "extracted_text"
- adds a derivative generation mixin that you may use to create thumbnails

class GenericFile < ActiveFedora::Base
  include Hydra::Works::FileSetBehavior
  property :state, predicate: ActiveFedora::RDF::Fcrepo::Model.state, multiple: false do |index|
    index.as :symbol, :facetable
  end
end
We need to implement a FedoraMigrate::Mover that is aware of this mixin:

```ruby
module FedoraMigrate::Works
  class FileSetMover < FedoraMigrate::ObjectMover
    def migrate_content_datastreams
      super
      if target.is_a?(GenericFile) && (ds = source.datastreams['content'])
        ofile = target.build_original_file
        mover = FedoraMigrate::DatastreamMover.new(ds, ofile, options)
        target.original_file = ofile
        save
        report.content_datastreams << ContentDatastreamReport.new(ds.id, mover.migrate)
      end
    end
  end
end
```
Once the content DS is migrating to the original_file property, we can generate derivatives in the rake task, for example:

```ruby
source = FedoraMigrate.source.connection.find(pid)
target = nil
options = { convert: "descMetadata" }  
mover = FedoraMigrate::Works::FileSetMover.new(source, target, options)
mover.migrate

target = mover.target
mover = FedoraMigrate::RelsExtDatastreamMover.new(source, target).migrate

target.create_derivatives if target.is_a?(GenericFile)
```

Be advised that this is somewhat slow- you may want to restrict the migration to a single object for expediency.
PCDM via Hydra-Works

With suitable libraries installed, Hydra-Works can create derivatives for more than images - but it requires characterization:

```ruby
source = FedoraMigrate.source.connection.find(pid)
target = nil
options = { convert: "descMetadata" }
mover = FedoraMigrate::Works::FileSetMover.new(source, target, options)
mover.migrate

target = mover.target
mover = FedoraMigrate::RelsExtDatastreamMover.new(source, target).migrate
if target.is_a?(GenericFile)
  Hydra::Works::CharacterizationService.run(target)
  target.save
  target.create_derivatives
end
```
PCDM via Hydra-Works

The characterization service does basic format analysis via FITS, and adds some technical metadata to our FileSet objects based on original_file.

<table>
<thead>
<tr>
<th>ns001: state</th>
<th>info:fedora/fedora-system:def/model#Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns003: hasFile</td>
<td><a href="http://localhost:8080/fcrepo/rest/dev/6/files/586e">http://localhost:8080/fcrepo/rest/dev/6/files/586e</a>;</td>
</tr>
<tr>
<td>ns005: exifVersion</td>
<td>9.06</td>
</tr>
<tr>
<td>ns005: orientation</td>
<td>normal*</td>
</tr>
<tr>
<td>ns005: software</td>
<td>Adobe Photoshop 7.0</td>
</tr>
<tr>
<td>ns006: hashValue</td>
<td>b8a86a3750e1dfae7700cb6ffdbe6638</td>
</tr>
<tr>
<td>ns007: compressionScheme</td>
<td>JPEG (old-style)</td>
</tr>
<tr>
<td>premis: hasCreatingApplicationVersion</td>
<td>0.6.2</td>
</tr>
<tr>
<td>premis: hasFormatName</td>
<td>JPEG File Interchange Format</td>
</tr>
</tbody>
</table>
Hydra::Works::WorkBehavior implements ordered versions of membership properties: ordered_members, and filtered accessors like ordered_file_sets & ordered_works

class Work < ActiveFedora::Base
  include Hydra::Works::WorkBehavior
  property :state, predicate: ActiveFedora::RDF::Fcrepo::Model::state, multiple: false do |index|
    index.as :symbol, :facetable
  end
end
The sample FCR 3 Work objects have ordered lists in a METS structMap, stored in a datastream called 'structMetadata'. For the membership to reflect this order, we need a new FedoraMigrate::Mover implementation.

class Work < ActiveFedora::Base
  include Hydra::Works::WorkBehavior
  property :state, predicate: ActiveFedora::RDF::Fcrepo::Model::state, multiple: false do |index|
    index.as :symbol, :facetable
  end
end
module FedoraMigrate
    module Works
        class StructureMover < FedoraMigrate::Mover
            def migrate
                before_structure_migration
                migrate_struct_metadata
                after_structure_migration
                save
                super
            end

            def migrate_struct_metadata
                ds = source.datastreams['structMetadata']
                if ds
                    ns = {mets: "http://www.loc.gov/METS/"}
                    structMetadata = Nokogiri::XML(ds.content)
                    members = {}
                    structMetadata.xpath("/mets:structMap/mets:div", ns).each do |node|
                        members[node['ORDER']] = node['CONTENTIDS']
                    end
                    members.keys.sort {|a,b| a.to_i <=> b.to_i}.each do |key|
                        member_id = id_component(members[key])
                        member = ActiveFedora::Base.find(member_id)
                        target.ordered_members << member
                    end
                end
            end

            def migrate_object(fc3_uri)
                RDF::URI.new(ActiveFedora::Base.id_to_uri(id_component(fc3_uri)))
            end
        end
    end
end

PCDM via Hydra-Works
class FedoraMigrate::Works::StructureMover < FedoraMigrate::Mover
  def migrate; ... end
  def migrate_struct_metadata
    ds = source.datastreams['structMetadata']
    if ds
      ns = {mets: "http://www.loc.gov/METS/"}
      structMetadata = Nokogiri::XML(ds.content)
      members = {}
      structMetadata.xpath("/mets:structMap/mets:div", ns).each do |node|
        members[node['ORDER']] = node['CONTENTIDS']
      end
      members.keys.sort{|a,b| a.to_i <=> b.to_i}.each do |key|
        member_id = id_component(members[key])
        member = ActiveFedora::Base.find(member_id)
        target.ordered_members << member
      end
    end
  end
end
class FedoraMigrate::Works::StructureMover < FedoraMigrate::Mover
  def migrate; ... end
  def migrate_struct_metadata; ... end
  # borrowed from FedoraMigrate::RelsExtDatastreamMover
  def migrate_object(fc3_uri)
    id_comp = id_component(fc3_uri)
    base_uri = ActiveFedora::Base.id_to_uri(id_comp)
    RDF::URI.new(base_uri)
  end
end

PCDM via Hydra-Works
With the mover implemented, you can add it to the migration in the rake task (remember to stub the hooks as well):

```ruby
if target.is_a?(GenericFile)
    Hydra::Works::CharacterizationService.run(target)
    target.save
    target.create_derivatives
end

if target.is_a?(Work)
    FedoraMigrate::Works::StructureMover.new(source, target, options).migrate
end
```
After running "rake clean" and "rake migrate", you should now see different contained resources for the works:

Constitution of the United States

UUID
Created at 2016-03-05T19:20:44.494Z by fedoraAdmin
Last Modified at 2016-03-05T19:20:58.91Z by fedoraAdmin
Children
PCDM via Hydra-Works

Checkpoint branch:
fedora-migrate-workshop/migrate-works

uses Hydra::Works to order the FileSets belonging to a Work via Proxies in DirectContainers

edits lib/tasks/migrate.rake to create derivatives of GenericFiles with the FileSetBehavior mixin
Questions? Ideas?

● freenode#projecthydra

● @barmintor
● armintor@gmail.com  /  ba2213@columbia.edu