Demonstration Objects

This document describes the demonstration objects that are distributed with Fedora.

After installing Fedora, you'll find these objects, in several formats, in your **`$FEDORA_HOME/client/demo`** `directory.`

On this page:

Ingesting the Demo Objects

These objects can be ingested into the repository in one of two ways.

- Using the Fedora Administrator GUI, selecting File, Ingest, Multiple Objects, From Directory and pointing to the demo/foxml directory.
- Running the fedora-ingest-demos command line script.

Once ingested, the demo objects can be viewed in a web browser using API-A-LITE. For example, to view the **demo:5** object:

```
http://localhost:8080/fedora/objects/demo:5
```

All demo objects are intended to work when the Fedora repository server is in a stand-alone condition (e.g., if the repository is running without a network connection, or if the repository is behind a firewall and not set up to receive outside connections)

Simple Document Demo

The Fedora data object **demo:18** demonstrates the simplest Fedora digital object scenario. It is the case where we aggregate content in the Fedora object, and let Fedora's default object behaviors provide access to the content. This is an example of a Fedora digital object that only has default dissemination services. In this case, there are 3 datastreams in the object, one for each format of a particular document (in this case the Fedora paper presented at ECDL2002). We can use the basic Fedora object dissemination service (also called "datastream disseminations") which are part of the basic content model shared by all objects. The basic content model is dynamically associated with every object in the repository (though it may optionally be statically associated). It has a default service definition (sDep) which provides basic operations for every object which includes the ability to list items in the object, get an item, get the dissemination index, get the Dublin Core record, and retrieve other information about the object. The results of these operations can be returned as either HTML (method names begin with "view...") or XML (method names begin with "get..."). The end result is that the object is simply a container for content and metadata. The user can view the contents and get any item from the object. While this scenario may be easy to implement and useful, it does not take advantage of Fedora's extensible service features where custom operations can be associated with an object.

Formatting Objects Demo

There are two demonstrations of using Fedora to display XML content styled using XSL Formatting Objects. First, the Fedora data object **demo:21** shows the transformation of native formatting object document stored as an inlined XML datastream into PDF. Second, the Fedora data object **demo:26** shows the use of formatting objects to process TEI documents.

Simple Image Demo

The Fedora data object **demo:5** demonstrates the UVA Simple Image behaviors by associating a simple dissemination with the object through its content model. There are 4 Datastreams in the object, one for each of four different image resolutions. The object is linked to one dissemination service which provides four behavior methods: getVeryHigh, getHigh, getMedium, and getThumbnail. The fulfillment of the service contract entails the Fedora HTTP Image Getter resolving the URL of the appropriate datastream for each of the UVA Simple Image behaviors. There are no transformations performed on the datastreams. This object shows how a service definition can be used to create a normalized set of methods for a particular type of object, an image object in this case, which is defined by a content model. The idea here is that the Simple Image service definition provides a standard set of dissemination services that can be used on any image object that conforms to the standard image content model. As we will see later, different variants of image objects can subscribe to the same service definition, and in some cases the datastreams will be dynamically transformed by a service to provide the appropriate image disseminations. This demo shows a simple one-to-one mapping of the datastreams in the object to the behavior methods.

Document Transformation Demo

The Fedora data object **demo:14** demonstrates the Document Transformation behaviors. There are 3 datastreams in the object, one XML source document, and two XSLT stylesheets. The object's content model provides one dissemination service which is associated with the "Document Transform" service definition and the Fedora Local Saxon Service (service deployment). Two services are available: getDocumentStyle1 and getDocumentStyle2. When these methods are run the repository mediates access to the Fedora Local Saxon Service to produce the appropriate transformation on the XML source in the object. The dissemination result will be one of two document styles.

Image Collection Demo

This demo illustrates the use of the Resource Index search service to fulfill collection behaviors.

For this demo to work, the Resource Index must be enabled prior to ingesting these objects.

A series of data objects (demo:SmileyBucket, demo:SmileyKeychain, etc.) subscribe to the image behaviors defined by the sDef object demo:DualResImage. Each of these image objects also use the RELS-EXT datastream to assert its membership in the demo:SmileyStuff collection. The demo:SmileyStuff collection subscribes to sDef object demo:Collection, which defines two methods: list and view. The collection object uses the demo:DualResImageCollection sDep to implement those behaviors.

To see the dynamic HTML listing of collection members in action, you can visit:

```
http://hostname/fedora/get/demo:SmileyStuff/demo:Collection/view
```

This dissemination first requests the list of members of the demo:SmileyStuff collection using the local risearch service. Then it uses the local saxon service to transform the XML results into a human-readable HTML page. The query text and the stylesheet are both datastreams of the SmileyStuff collection and act as inputs to the list and view behaviors, respectively.