Tutorial 2 - Creating Fedora Objects

Fedora Tutorial #2
Getting Started:
Creating Fedora Objects using the
Content Model Architecture
Fedora 3.0
July 23, 2008

Author: The Fedora Development Team

Purpose: This tutorial introduces the basic development questions, design concepts and project goals of the Flexible Extensible Digital Object Repository Architecture (Fedora).

Audience: This tutorial is intended for repository administrators or content developers who will be using the Fedora software.

Table of Contents

Figures

What is This Document and Who Should Read It?

This is an introduction for system developers and repository managers who are new to the Fedora Repository open-source content management software. This is a hands-on tutorial. It assumes that you have already installed the Fedora software and are at a computer with access to a Fedora repository through the `Fedora Administrator` while reading this tutorial.

You don't have to have to be a programmer to understand and use this tutorial. However, you should be familiar with the operation and structure of web servers and web services.

This document is *not* intended for end users of content disseminated by a Fedora repository.

What is Fedora and What Does It Do?

Fedora is content management software that runs as a web service within an Apache Tomcat web server. Fedora provides the tools and interfaces for creation, ingest, management, and dissemination of content stored within a repository. There are a number of features that distinguish Fedora:

1. It supports the creation and management of digital content objects (from this point on called a *Fedora Digital Objects* or *FDO*) that can aggregate data from multiple sources. For example, a *FDO* might be a set of *TIFF* images that are the individual page images of a scanned document. The data sources may be either locally managed within the Fedora software or sourced from another URL accessible network server. The data sources may be content or metadata. You may think of these *FDOs* as advanced digital *documents*, especially in light of the feature described next.
2. It supports the association of web services with the *FDOs*. These services typically consume the data packaged within the *FDOs* to produce dynamic disseminations from them. For example, the *FDO* described above with multiple *TIFF* page images may be associated with a service that OCRs the images that are components of the *FDO* and disseminates an *HTML* version of the pages. The services may be either local to the machine of the respective Fedora server or sourced from another network accessible server that is addressable via a URL. In this manner, Fedora acts as a mediation layer that coordinates local and distributed data and web services within a uniform framework. This is illustrated in Figure 1.
3. It provides uniform access web-based interfaces to these *FDOs*, through REST requests and more powerful SOAP-based methods. These interfaces consist of a set of built-in methods to access characteristics common to all *FDOs* such as key metadata and internal structure. These include a method to introspect on an object to reveal the set of methods that constitute the extended behavior of that object. For example, a client could use these built-in methods to "learn" about the capability of the *FDO* described above to dynamically disseminate an *HTML* page from a set of *TIFF* images.
   The benefits of these are two-fold:
   a. Clients accessing Fedora Digital Objects can rely on uniform access regardless of the nature of the object.
   b. The disseminations available from an object are independent of the internal structure of the object. For example, the client interface of the example above in which *HTML* is disseminated from a set of source *TIFF* pages could remain constant regardless of whether the underlying object contained *TIFF* images, *JPEG*, *PDF*, or even simple static *HTML*. This gives the content developer great freedom to modify a repository's internals without disrupting the client and user views of the content.
4. It presents a uniform and powerful REST and SOAP-based management interface. All internal operations of the repository such as object creation and management are available through these APIs, providing the hooks for integrating Fedora into a variety of environments. These makes Fedora useful as the foundation for advanced content management applications.
5. It includes a comprehensive versioning framework that tracks the evolution of objects and provides access to earlier versions.
6. It includes a basic relationship framework for representing the links among *FDOs*.
7. It supports ingest and export of *FDOs* in a variety of *XML* formats. This enables interchange between Fedora and other *XML*-based applications and facilitates archiving tasks.

A number of these features are illustrated in Figure 1.

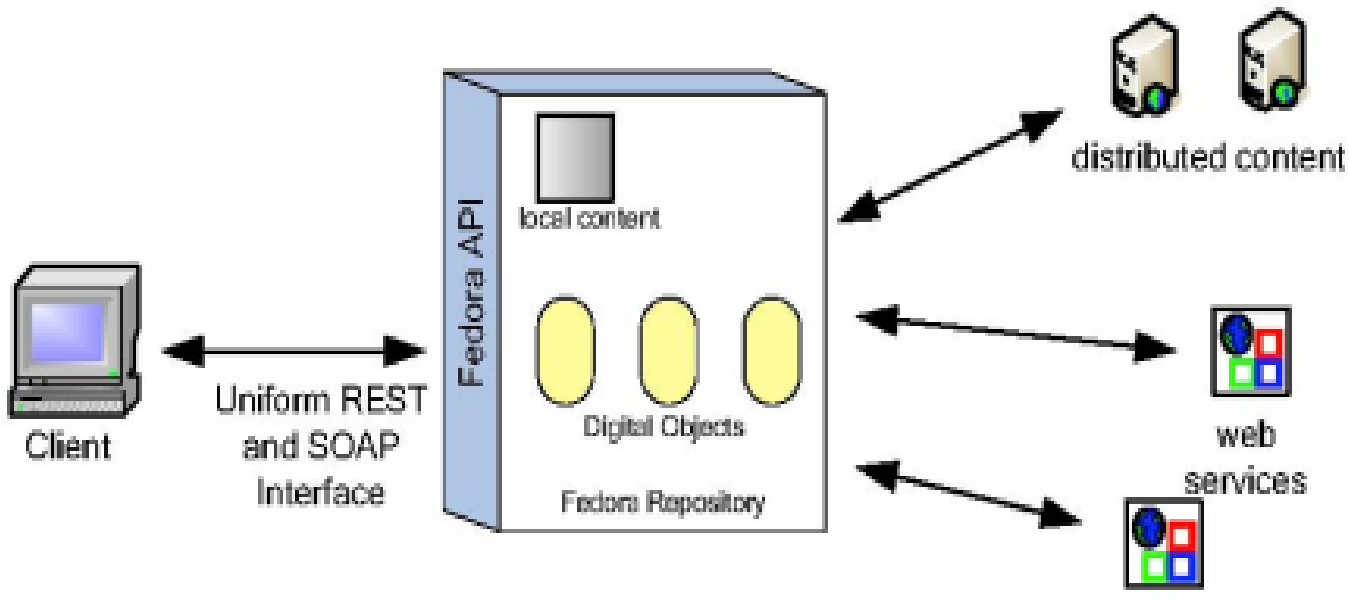


Figure 1 - Fedora repository as mediator for services and content

Why Should You Use Fedora?

Fedora may be the wrong choice for management of simple static web pages. There are a number of excellent tools for *HTML* editing and web site creation. Fedora is more appropriate for more advanced content management tasks. These include management of content and associated metadata, multiple versions of content, content available in multiple formats, and dynamically generated content from local and dynamic sources.

How Should You Read This Document?

This document is intended to be hands-on, with you trying the examples on a running Fedora repository. You should therefore, have already downloaded and installed Fedora, and started a server. You should then access the Fedora repository by running the Fedora Administrator interface, `fedora-admin`, which is located in the `FEDORA_HOME/client directory` (you can start this program from the command line if you have configured your environment variables properly). Upon starting up the administrator interface you will be presented with the `Login` screen shown in Figure 2. This document assumes that you have not changed any of the configuration defaults for your Fedora server so the password you enter should be **fedoraAdmin**. If you have changed your configuration values or are running the Fedora Administrator from a machine different from the machine on which your Fedora server is running you will need to change the values in the `Login` screen appropriately.

Figure 2 – Fedora Administrator Login Screen

You should read this document in order, since later examples assume knowledge of techniques and definitions introduced earlier.

Conventions Used in This Document

The font conventions used are:

1. *Defined terms are introduced like this.*
2. **Text in dialog boxes and windows is shown like this.**
3. `URLs, directory paths, file names, and similar items are shown like this.`

All path names assume that you have set your `FEDORA_HOME` environment variable and descend from the directory defined by that variable.

All URLs that access the Fedora repository assume that the `host:port` of the repository is `localhost:8080`.

Getting Started: Using Fedora for Aggregating Content

This section describes how to create digital objects in Fedora that aggregate data from multiple sources. The examples demonstrate how to do this with both local data and data from networked sources. This section provides the foundation for the next section, which describes how to use Fedora to create dynamic content by exploiting web services. Make sure you understand the basic concepts here, before moving on to that next section

Some Basic Definitions

To understand content aggregation in Fedora, you need to be comfortable with two terms:

1. *Fedora Digital Object* or *FDO* – This is the basic unit for information aggregation in Fedora. At a minimum a *FDO* has:
    a. A *Persistent Identifier* or *PID* – The *PID* provides the key by which the *FDO* is accessed from the repository.
    b. *Dublin Core* – It provides a basic description of the *FDO*.
2. *Datastream* – A component of a *FDO* that represents a data source. A *FDO* may have just the basic *Dublin Core Datastream*, or any number of additional *Datastreams*. Each *Datastream* can be any *MIME*-typed data or metadata, and can either be content managed locally in the Fedora repository or by some external data source (and referenced by a URL). When you create a new *Datastream* in a *FDO*, you assign it to one of four types, or *control groups*, depending on the nature of the data that it represents.
    a. *Managed Content (M)*: *Datastream* content is stored and managed within the Fedora repository's persistent storage. The content can be any *MIME* type including *XML*.
    b. *Inline XML (X)*: A special case of M, restricted to well-formed *XML*. In this case, the *Datastream* content is stored as part of the *XML* structure of the *FDO* itself and is thus included when the it is exported (e.g., for archival purposes).
    c. *Externally Referenced (E)*: *Datastream* content is external to the Fedora repository and is referenced by a URL that is recorded within the *FDO*. The content can be any *MIME* type including *XML*.
    d. *Redirected Content (R)*: Like E, but *Datastream* content is delivered to the client without any mediation by Fedora; i.e., via an *HTTP* redirect. You should use this *Datastream* type when the external content is a web page with relative links or it is streaming audio or video. The content can be any *MIME* type including *XML*.

Decisions about what to include in a *FDO* and how to configure its *Datastreams* are basic modeling choices as you develop your repository. The examples in this tutorial demonstrate some common models that you may find useful as you develop your application.

Example 1: Making a Document Available in Multiple Formats

It is often useful to provide access to a digital document in several formats. For example an ePrints server might provide *HTML* for those who wish to render the document in a browser, *PDF* for those who wish to view the document with author-determined formatting, and TeX for those who wish to access and use the document source. This example demonstrates how to construct a *FDO* where each *Datastream* corresponds to an available format. More advanced techniques, demonstrated later in this tutorial, make it possible to achieve the same results by generating formats dynamically from a single base format. But for now, we'll stick to simple static aggregation.

Start by selecting **File/New/Data Object** in the **Fedora Admin GUI**. Complete the **New Object** dialog box as shown in Figure 3.

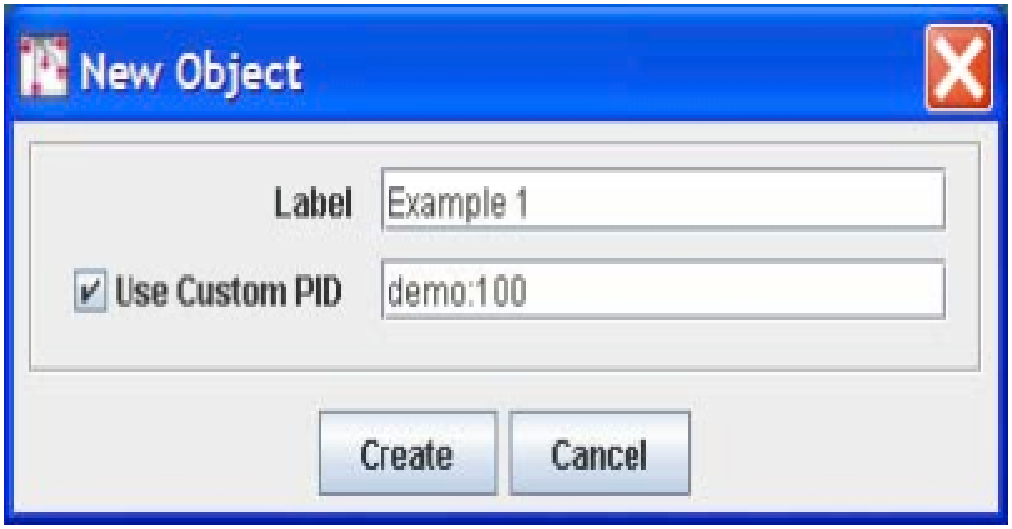


Figure 3 – New Object Dialog

Check the box for **Use Custom PID** and enter `demo:100`. Note that when you do not assign your own *PID*, the Fedora repository will create one for you. Select the **Create** button and you should see a window like that shown in Figure 4. Observe that the *PID* of the created object (in this case `demo:100`) is displayed in the title bar.

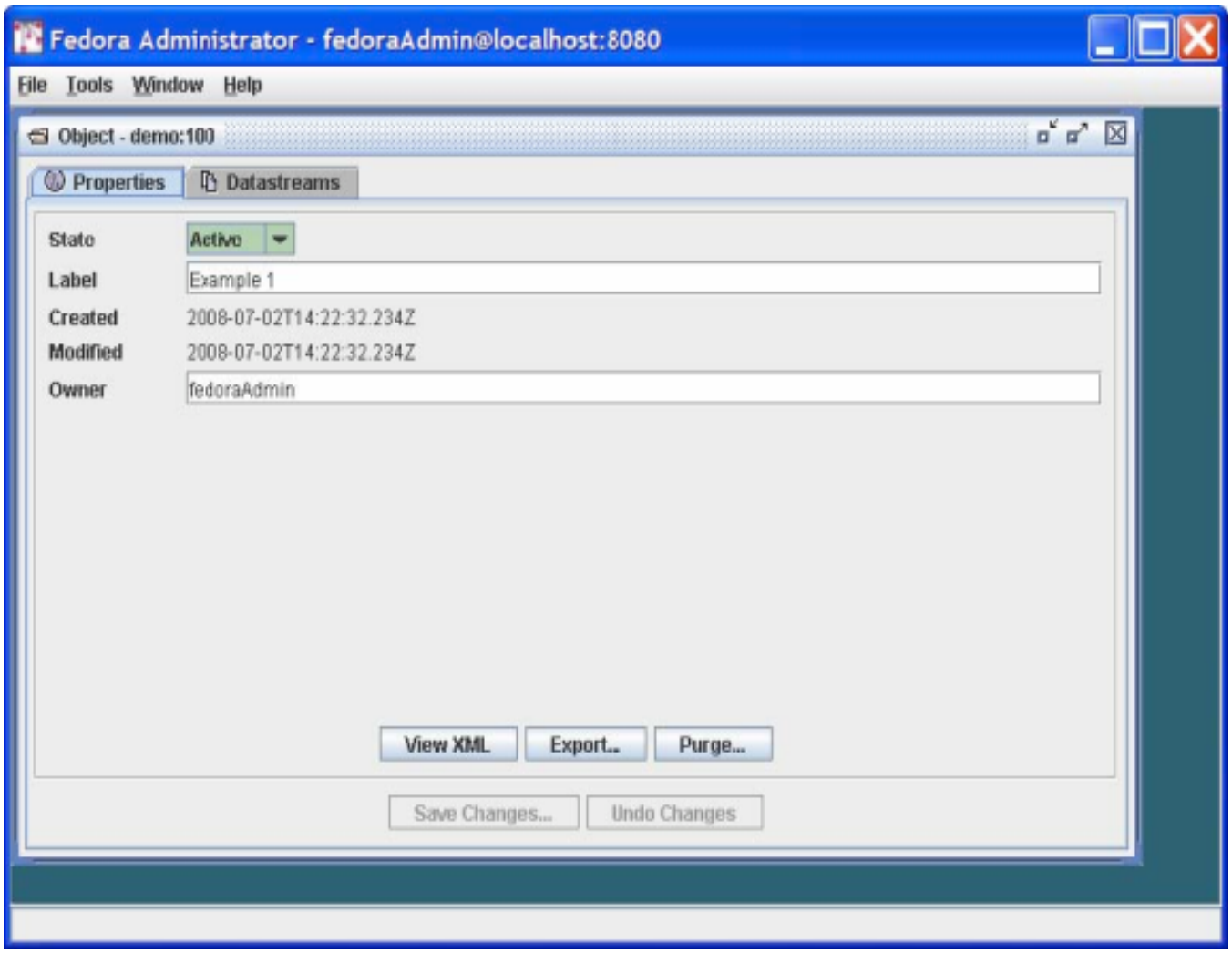


Figure 4 – Configuring an Object

Since our task here is to define the *Datastreams* in the object, click on the *Datastreams* tab and you will see a window like that shown in Figure 5. Note that at this point there is only one *Datastream* in the object – the DC *Datastream* containing basic descriptive metadata that was automatically created by Fedora. You can select that *Datastream* and select the `Edit` button to see the its default contents, with the DC title and identifier fields already filled in.

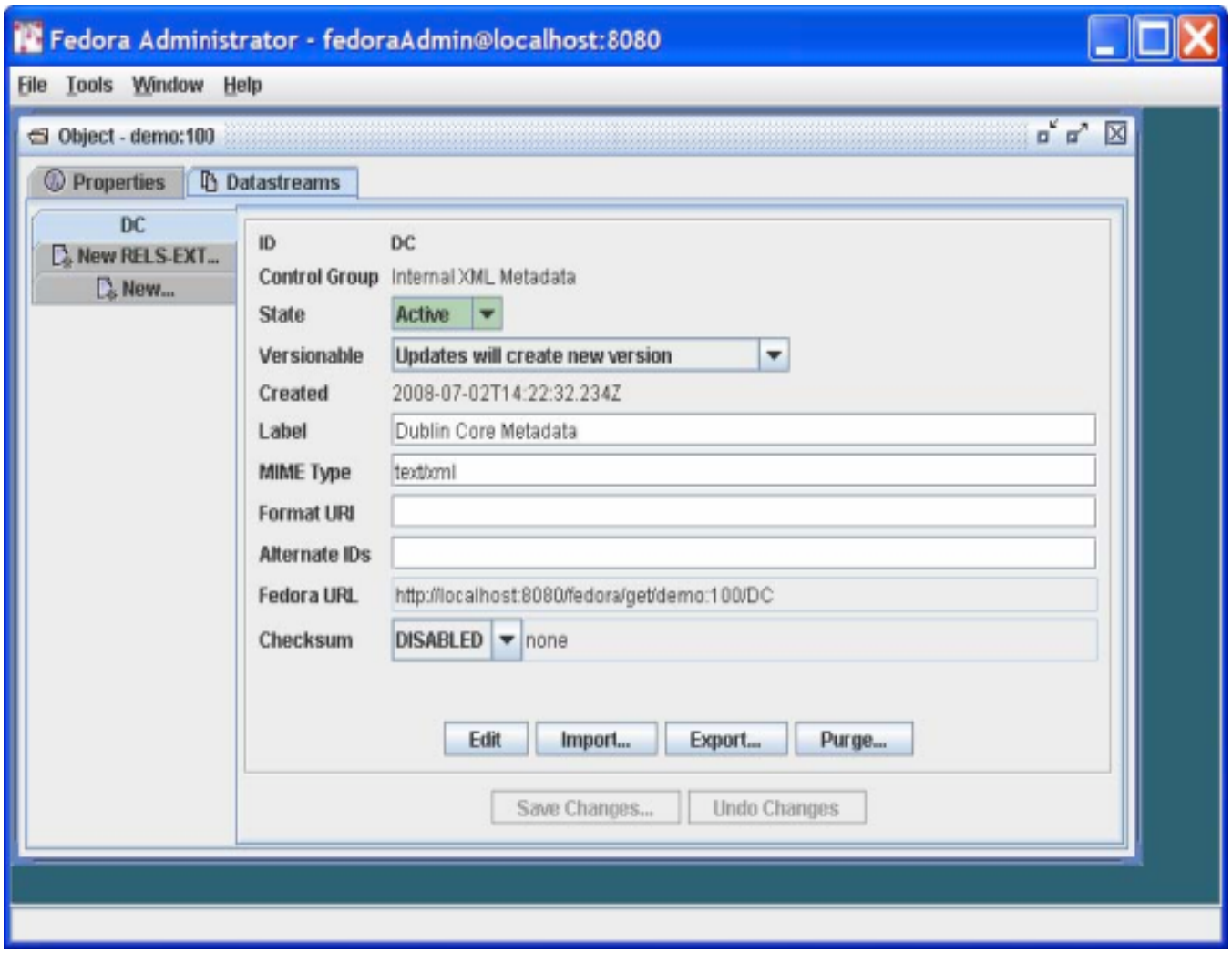


Figure 5 – *Datastream* Display

A few points to note about what you have done so far:

1. You will notice that the `Control Group` of the DC *Datastream* is `Internal XML Metadata`. As explained earlier, Fedora has a number of control group types, of which this is one. This type is appropriate for metadata that is represented in *XML – Dublin Core* metadata being one example. A *FDO* can have multiple metadata *Datastreams*, for example *MARC*, *LOM*, *Dublin Core*, and others.
2. You can directly edit the Dublin Core metadata – e.g., add new Dublin Core fields – by selecting the `Edit` button and modifying the contents of the text pane. When you press `Save Changes`..., Fedora will check that the *Datastream* is well-formed *XML*.

You may also create Dublin Core metadata (or any other *XML*-based metadata) in an external *XML* editor and using the `Import`… button to replace the *Datastream* with this data. When you press `Save Changes`..., Fedora will check that the *Datastream* is well-formed *XML*.

You will notice that there are optional fields on the *Datastreams* pane for `Format URI` (to refine the media type meaning with a URI that more precisely identifies the media type) and **`Alternate Ids`** to capture any other existing identifiers you would like to associate with a *Datastream*. We will not be using these in this tutorial.

It is now time to add the ePrints document formats as new *Datastreams*. You can find content for creating the *Datastreams* for this example in:

- `FEDORA_HOME/userdocs/tutorials/2/example1/artex.html`
- `FEDORA_HOME/userdocs/tutorials/2/example1/artex.pdf`
- `FEDORA_HOME/userdocs/tutorials/2/example1/artex.tex`

NOTE : Tutorial files are no longer included with Fedora. You can retrieve the needed files from Fedora 3.0 at sourceforge.

To do this, select the `New`… tab on the left side of the *Datastreams* window. We'll start with the `text/html` format. To insert data into the *Datastream*, you use the `Import`… button. This presents a dialog that will allow you to import from your local file system or from a URL.

Your completed *HTML Datastream* should look like the dialog as shown in Figure 6 (after you have imported the content).

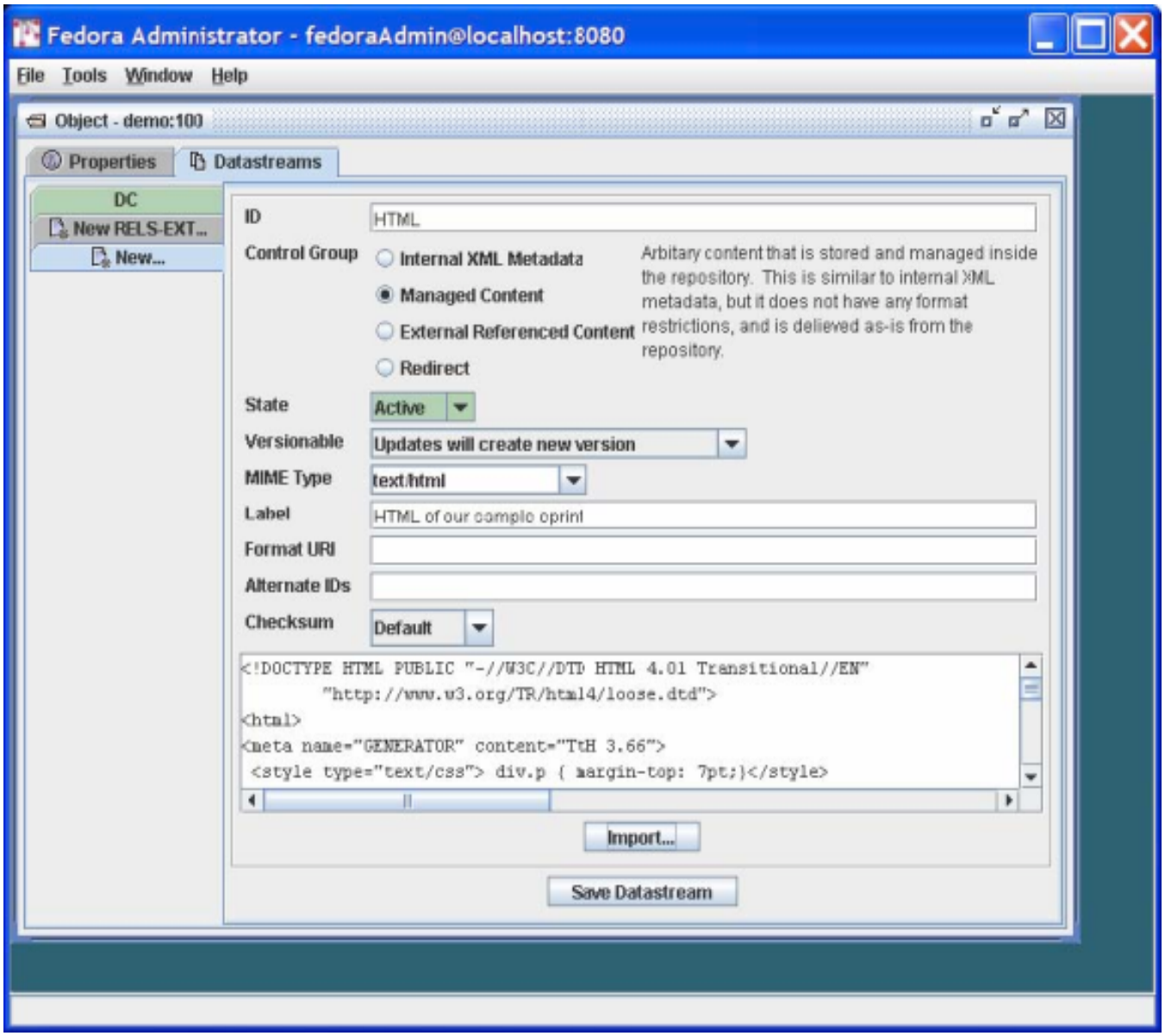


Figure 6 – Adding a New Managed Content *Datastream*

A few notes on the contents of this dialog:

1. The `ID` of the *Datastream* should be a single token. By convention, it describes the purpose of the *Datastream*.
2. The `Label` can be a longer, more descriptive string.
3. Note that the `Control Group` is `Managed Content`. As shown in the descriptive text this *Datastream* type is appropriate for any type of data (*MIME* type), in contrast to `Internal XML Metadata`. Once you select this radio button, you can select from the variety of `MIME Types` of the managed content – in this case `text/html`.

You can now select the `Save Datastream` button and repeat the same process to add the *PDF* and *TeX Datastreams*. For the *PDF*, you can select *MIME* Type: `application/pdf` and import the file `ex1.pdf`. For TeX, you can select *MIME* Type: `text/plain` and import the file `ex1.tex`. In each case you should enter appropriate IDs and Labels.

You're done! Your *Datastreams* window should now look something like that shown in Figure 7, showing all the *Datastreams* you have entered in the left-side tabs in the window.

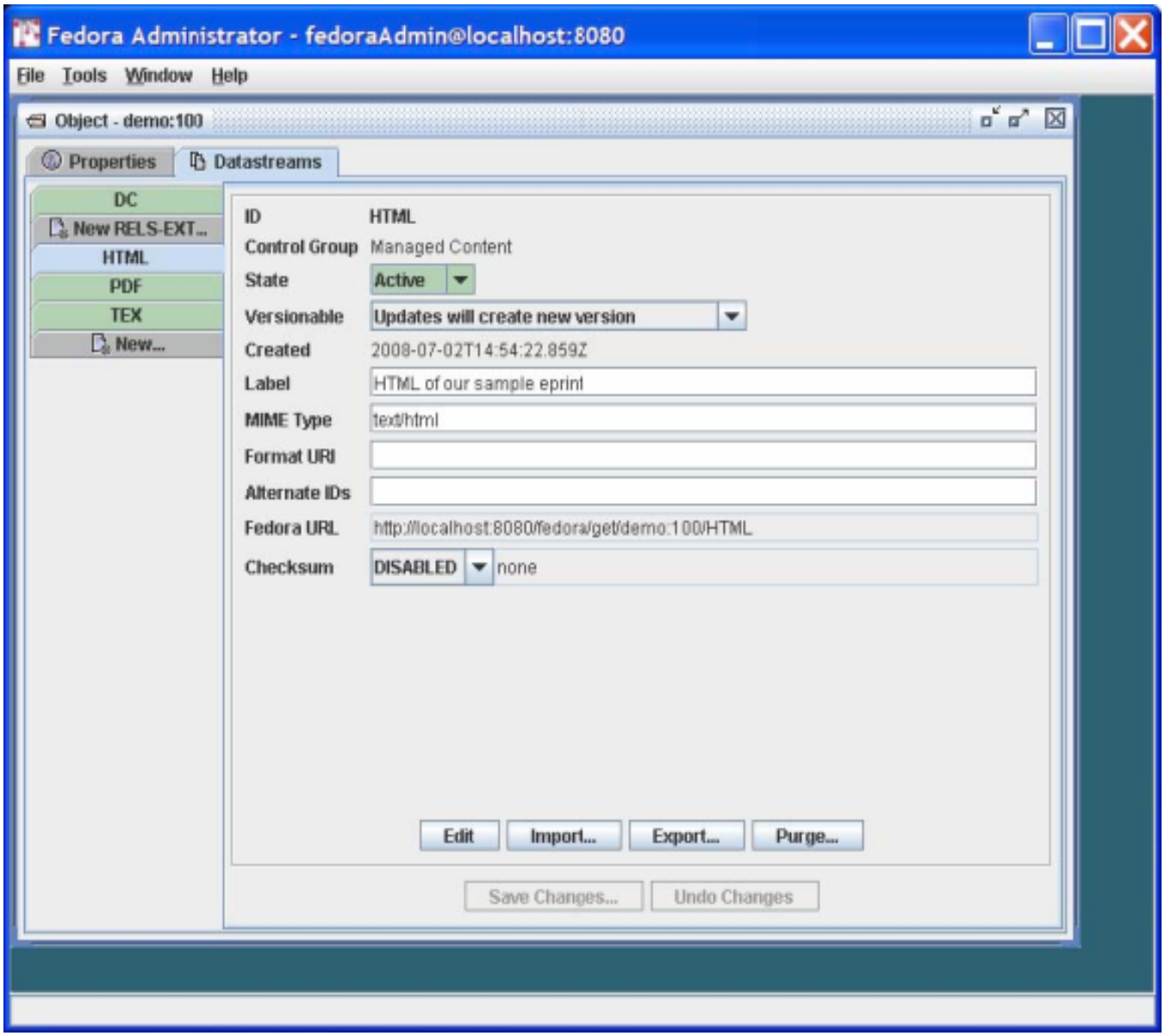


Figure 7 – Complete *Datastreams* for Example 1

You will notice as you click through each *Datastream* that there is a Fedora URL, giving the unique URL to access each *Datastream* from the Fedora repository. Try going to a browser and entering one of these URLs – the browser will download the *Datastream* and display it. These URLs can be used by web applications and REST-based web services that access *Datastreams* from Fedora Digital Objects. Note that if you are building SOAP-based web services, there are also SOAP methods (`getDataStream` and `getDatastreamDissemination`) that provide *Datastream* access. You can also try entering the root URL for the entire *FDO*, which is simply the common prefix of all the *Datastream URLs* – e.g., http://localhost:8080/fedora/get/demo:100. This accesses the header page for the *FDO*, which allows you to access its Datastreams (available through the item index hyperlink) and disseminations (available through the dissemination index hyperlink).

Figure 8 illustrates the structure of the object you have created and the correspondence of REST-based access requests to the object and its components (via API-A-LITE).

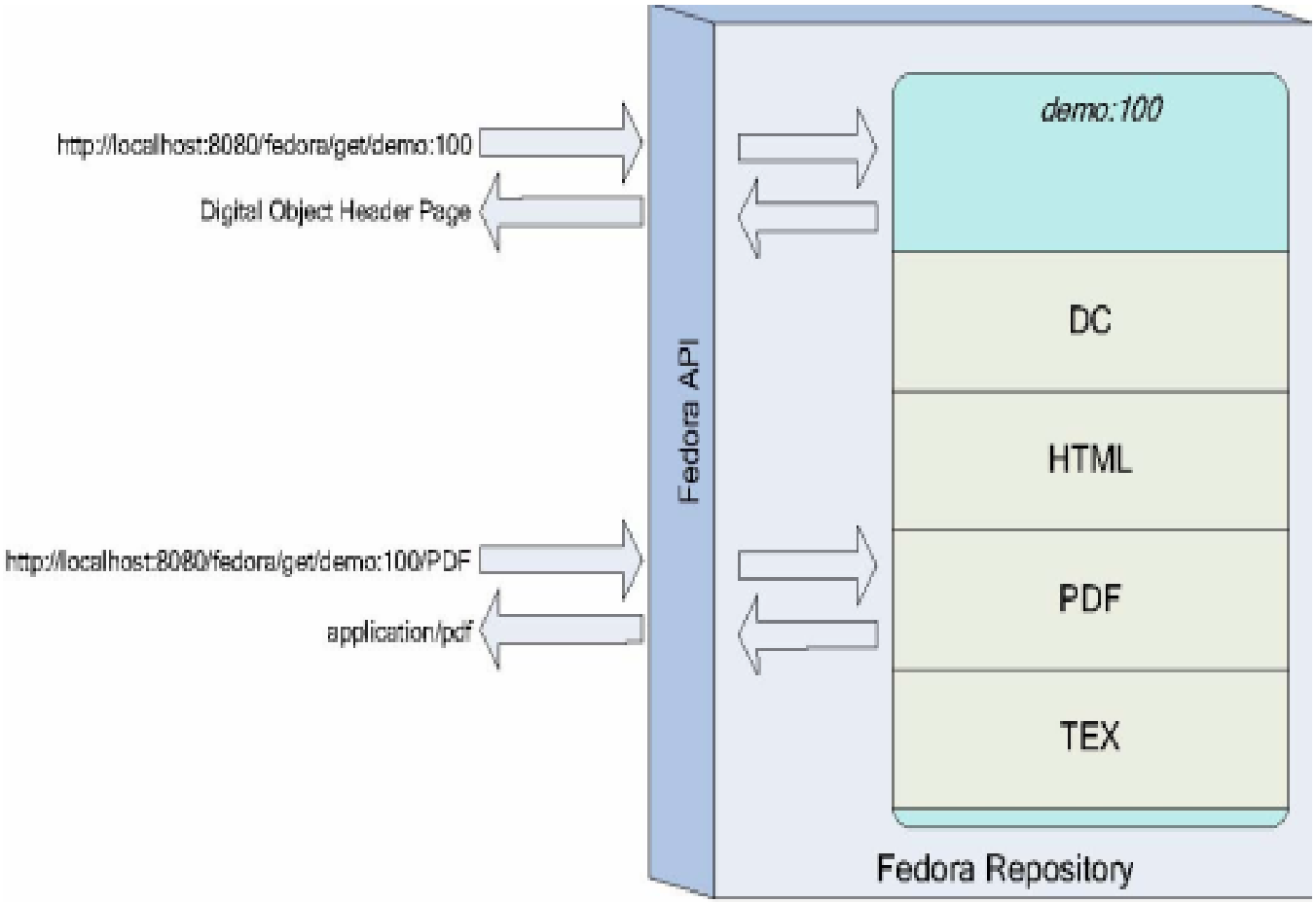


Figure 8 – Example 1 *Fedora Digital Object* and *Datastreams*

Example 2: Creating a Surrogate for Distributed Content

The previous example demonstrated how to aggregate imported content into a *Fedora Digital Object*. There are many reasons why importing content into a repository might not be appropriate such as rights restrictions or the dynamic nature of the content. To accommodate these restrictions, *FDOs* may contain Datastreams that reference externally managed content, and in fact may mix local and distributed data sources.

This section describes how to do this where the motivating example is the creation of a hypothetical learning object in an educational digital library, such as the NSDL}. The *FDO* created in this example combines three frog images from the NSDL collection and some locally-managed text.

To get started follow the same procedure as illustrated in Figure 3, this time entering *Example 2* as the Label and *demo:200* as the custom *PID*. As in Example 1, select the *Datastreams* tab and then enter the information as shown in Figure 9.

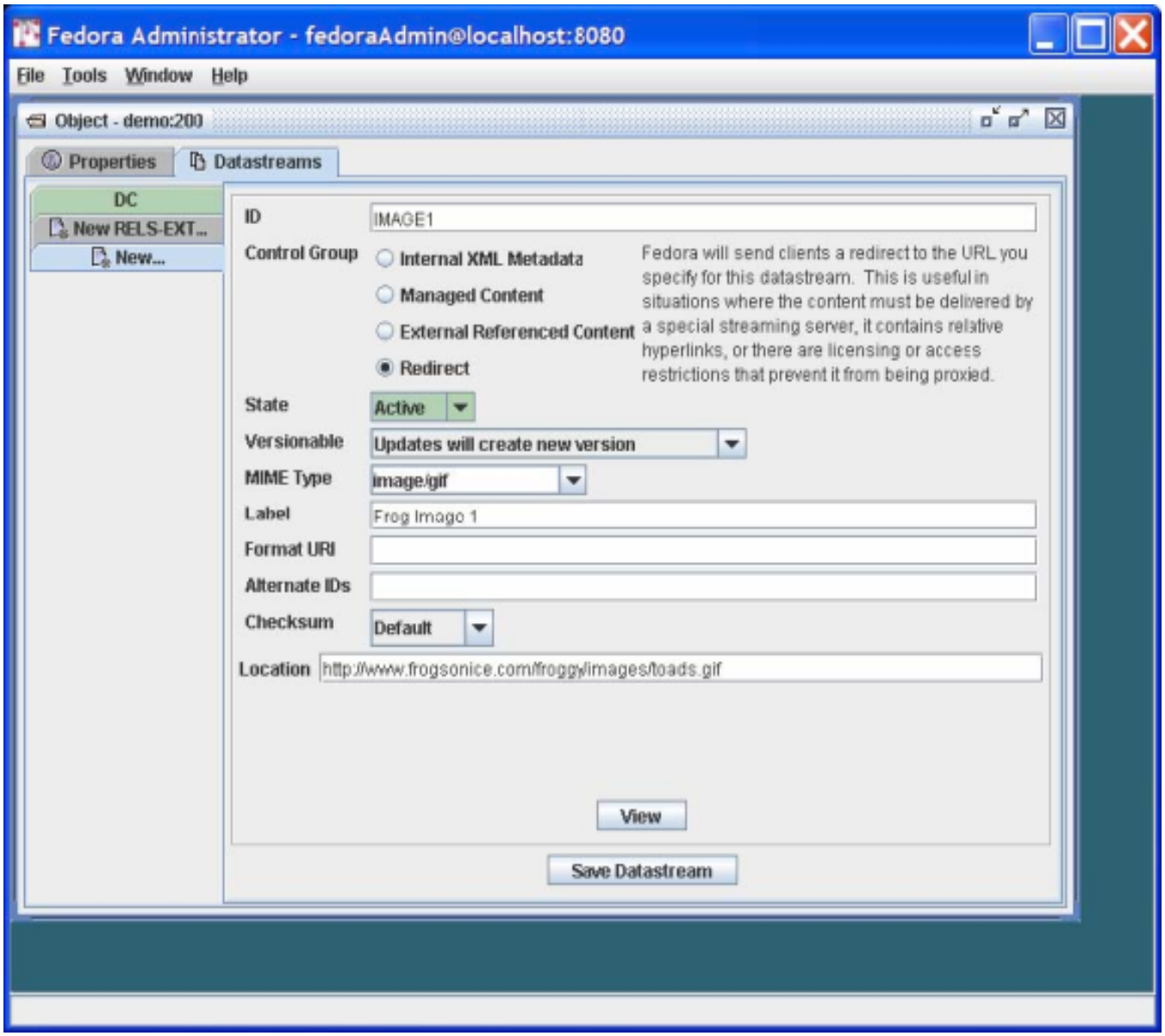


Figure 9 – Adding a *Datastream* with Type Redirect

You will enter the Datastream identifier of IMAGE1, a label for this Datastream, and then information about the content. The content is of *MIME* type `image/gif`. You should select the Control Group of Redirect, and then enter a URL that specifies the Location of the image file, specifically:

http://www.frogsonice.com/froggy/images/toads.gif

A few notes on the contents of this dialog:

- Pertaining to the selection of a Control Group, you have two choices if you want the Datastream to point to content that resides outside the Fedora repository (External Referenced Content *and* Redirect)**.**In this case we chose Redirect . To review, the meaning of the two options for mapping to external content are:
    - External Referenced Content is useful when you want Fedora to mediate access to the Datastream, for example when you want to hide the source URL from the user. Fedora mediates access to these Datastreams, meaning that the content is streamed through the Fedora server.
    - Redirect. makes use of a simple HTTP redirect to provide the content. This is useful when there are relative hyperlinks in the external content, but reveals the source URL to the user.
- Make sure that the *MIME* type choice matches that of the content offered by the external source, in this case image/gif.

In the same manner, you can now proceed to add the two other Datastreams with locations: http://www.werc.usgs.gov/fieldguide/images/hycafr.jpg and http://www.aquariumofpacific.org/images/olc/treefrog600.jpg.

You should respectively identify these Datastreams as IMAGE2 and IMAGE3. (Note that if these sample URLs are no longer active, you can enter other URLs pointing to *JPEG* images to complete this tutorial exercise.)

Finally, add another Datastream labeled MyText (containing some descriptive text about the images), with *MIME* type `text/html`. Assign this Datastream a Control Group of Managed Content indicating that the content will be imported and stored permanently in the Fedora repository. Import the content from the following location:

```
FEDORA_HOME/userdocs/tutorials/2/example2/mytext.html
```

The resulting Datastream window should now look like that shown in Figure 10.

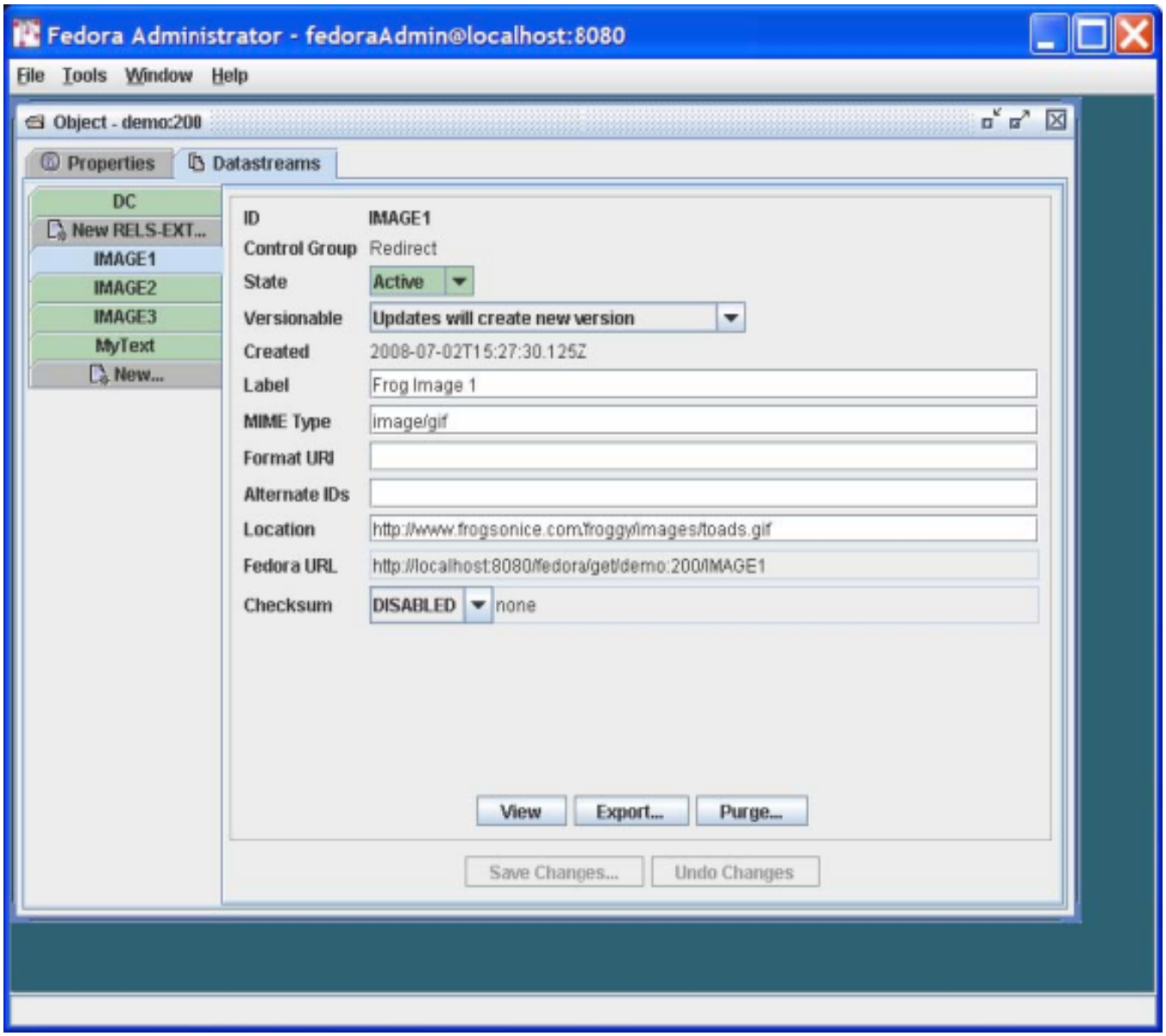


Figure 10 – Example 2 *Datastream* Display

You're done! Figure 11 illustrates the role of the redirected Datastream at the time of *FDO* access via the Fedora REST-based interface (API-A-LITE). You can see this by going to the *FDO* profile page at: http://localhost:8080/fedora/get/demo:200

You can access the Datastreams for this *FDO* by viewing the item linked to from the object profile page. Then, select the link for one of the redirected Datastreams. Fedora will redirect your browser to the location of the Datastream content, without streaming the content through the Fedora repository server.

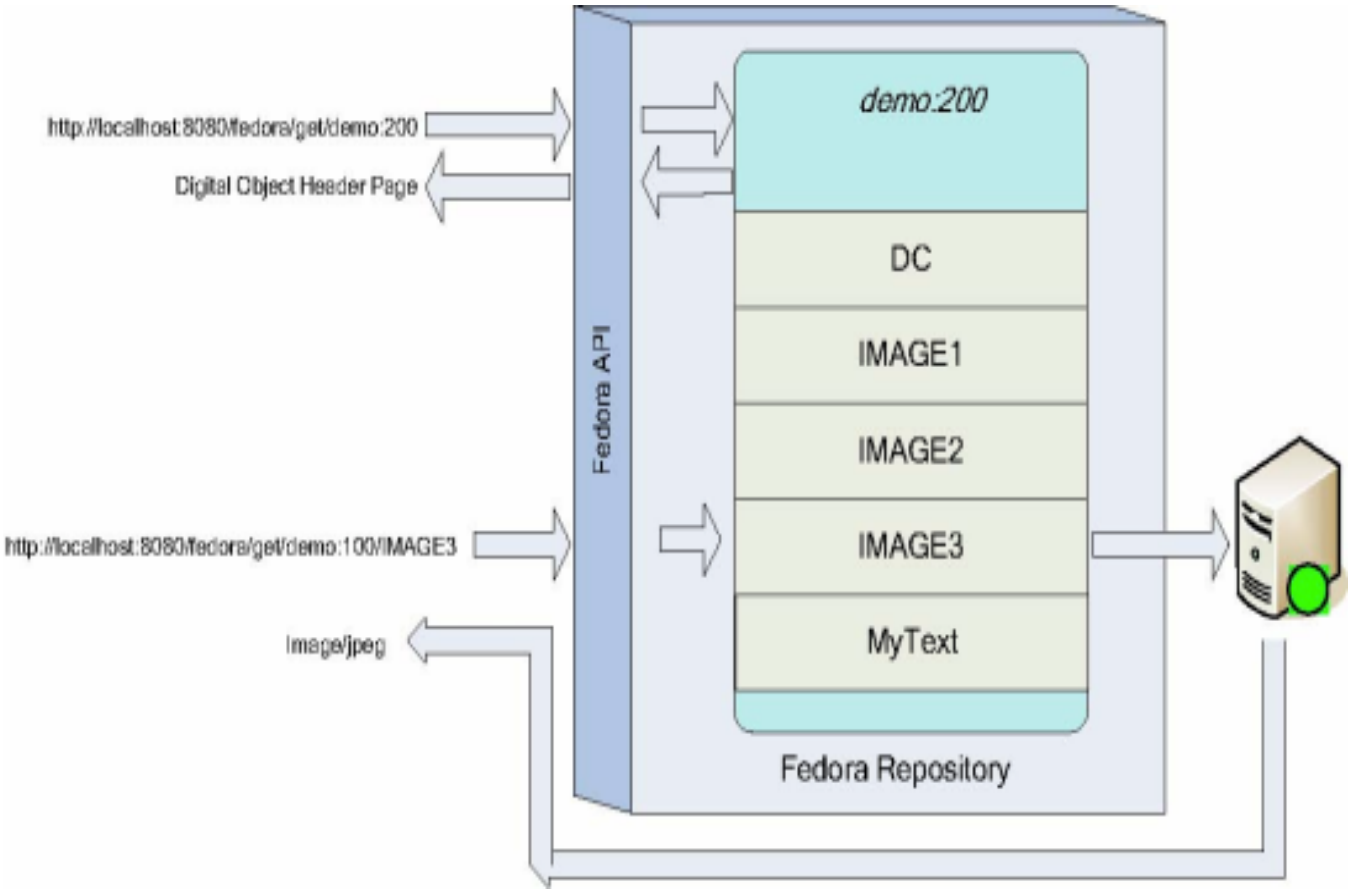


Figure 11 – Example *Fedora Digital Object* and Redirected *Datastream*

Using Fedora to Produce Dynamic Content

The examples described so far demonstrate the basic content aggregation features of Fedora. As mentioned already, the power of Fedora lies in its ability to associate the data in a *FDO* with Web services to produce dynamic disseminations. Some examples of this capability are as follows:

1. Rather than packaging multiple formats of a document as in Example 1, it is possible to have a *FDO* with one Datastream in a source format (e.g. TeX) and then associate a service with the *FDO* to transform the source format into multiple output formats (e.g. *PDF* and *HTML*). An obvious advantage of this is that any changes to the source format propagate out to the derived formats. Furthermore, less content is stored and/or duplicated in the repository.

- Rather than packing multiple metadata *XML*-based metadata formats in a *FDO*, it is possible to package a single base metadata format in a *FDO* (for example, fully qualified Dublin Core) and use that base format as the basis of metadata crosswalks. To do this, one could associate an *XSLT* engine (e.g. Saxon) service with the *FDO* that processes the base format with a transform *XSL* document (packaged as a Datastream in another *FDO*) to derive one or more additional formats.
  In both cases, static and dynamic, disseminations are available via REST or SOAP requests from clients to the Fedora Access service (API-A and API-A-LITE). The nature of the disseminated content – the format of the underlying data, where it is located, and whether it is static or dynamically generated – is invisible from the client perspective. As a result, a repository manager can significantly alter the nature of a *FDO* and the web services that it uses while maintaining the same interface vis-à-vis the client. Correspondingly, two *FDOs* with entirely different structure can appear the "same" from the perspective of consuming clients.

The remainder of this section presents a series of examples demonstrating how to create *FDOs* that exploit Web services. The initial examples make use of services available in the Fedora software release (they run as "local services" within the Fedora server container). Later examples demonstrate how to construct your own custom objects with external web services. Before proceeding with the examples, this introduction summarizes the concepts and defines the terms used in the examples. Don't worry if the concepts are not entirely clear at first. You should read them now and then refer back to them as you work through the examples.

Figure 12 shows an abstract view of the different components of the Fedora repository architecture that are key to how Fedora produces "disseminations" of *FDO* object content.

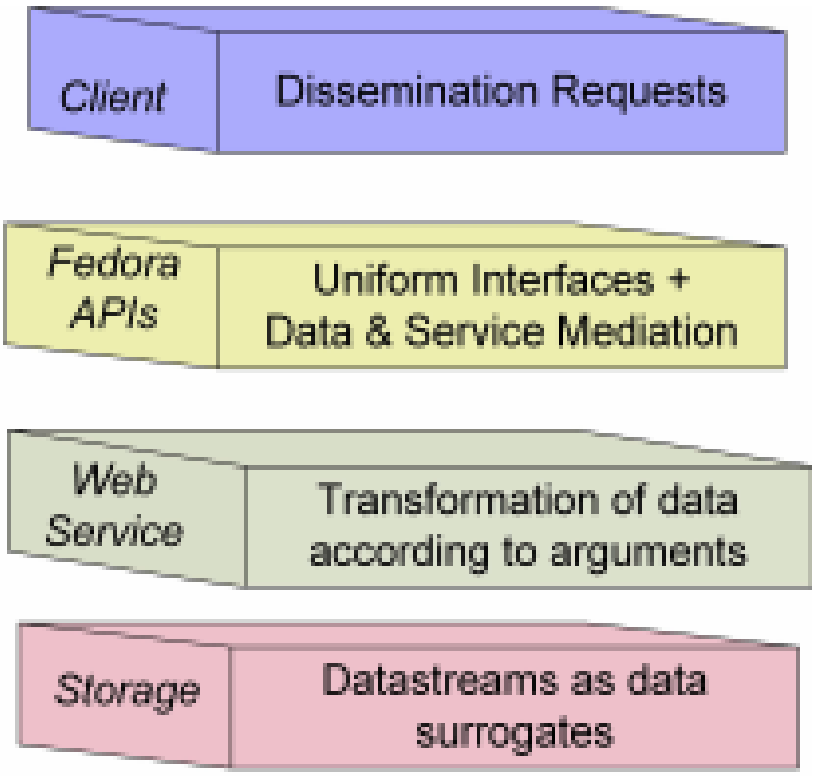


Figure 12 – Abstract View: Key Fedora Components for Producing Disseminations of Content

These layers are:

1. *Client*: Clients make requests for content disseminations through the Fedora Access service APIs (i.e., API-A-LITE and API-A). These interfaces include operations for discovering and accessing all disseminations that are available for a particular *FDO*. A *FDO* can have both static and dynamic disseminations, which are described below.
2. *Fedora APIs*: The Fedora repository service is exposed via a uniform set of APIs. Fedora's API-A and API-A-Lite provide operations (methods) for accessing *FDO* content. While the default mode of accessing a *FDO* delivers the Datastreams (i.e., repository returns the bitstream represented by the Datastream un-transformed), the *CMA* (*Content Model Architecture*) enables defining any number of *custom* services for accessing Datastream content. These custom services are produced when the Fedora repository service calls another Web service to transform Datastream content. Such transformations can be thought of "virtual" views of *FDO* content, since these views are created dynamically at runtime.
3. *Web Services*: These are Web-accessible programs that are invoked by HTTP to produce disseminations of *FDO* content. Note that the Fedora repository itself is a Web service to access the default services of *FDOs*. Also, Fedora can interact with other Web services to product custom access services that transform *FDO* content on-the-fly. In this tutorial we will describe how Fedora interacts with simple REST-based services to product such custom services. Custom services are produced when the Fedora repository service itself makes outbound service calls to other Web services using simple REST-based requests. We will not discuss Fedora interacting with SOAP-based web services here.
4. *Storage*: *FDOs* objects are stored by the Fedora repository service. Datastreams are constituent parts of *FDOs* – essentially metadata about the bytestreams. Fedora interacts with low-level storage to access *FDOs* to fulfill client requests for access to content. Datastreams capture the raw content. As shown in the previous examples, Datastreams can be directly disseminated via the Fedora Access service. Also, Datastreams can serve as input to other custom services that are produced on-the-fly when the Fedora repository service calls upon another Web service at run time (using a raw Datastream as input).

The process of creating *FDOs* with dynamic content disseminations involves creating linkages between these layers. During this process you will create and employ the following:

1. *Service Definition* (*SDef*): A *FDO* that is a template for client-side services, defining a set of abstract operations (methods) and their client-side arguments. Association of a *SDef* with a *FDO* augments the basic behavior of the object with the operations defined in the *SDef* template. A *SDef* may be associated with more than one *FDO*, thereby augmenting all of them with the same operations.
2. *Service Deployment* (*SDep*): A *FDO* that registers within Fedora the capability of web service(s) to perform the operations defined by a specific *SDef*. This registration includes defining service binding metadata encoded in the Web Service Description Language (*WSDL*) and also a *data profile* of the *SDep*. The data profile defines the types of inputs that are considered compatible with the service. In particular it declares the *MIME* types that are needed by the respective web service to perform its task. Multiple *SDeps* may be registered for an individual *SDef*, thereby exposing a generic client-side interface (defined by the *SDef*) over multiple data and web service foundations (defined by the *SDep*).
3. *Content Model* (*CModel*): A *FDO* that is used to store information which will allow Fedora to determine whether a data object, which asserts conformance to a content model, is valid. The Content Model is also important for performing disseminations in Fedora, based on the Content Model Architecture. A Data Object will indicate which Content model they represent via a special RELS-EXT relationship. The Content Model will in turn indicate which *SDef(s)* it is associated with (also with a special RELS-EXT relationship).

These three kinds of special Fedora objects are stored in Fedora repositories. The set of all *SDefs* represents a "registry" of all the kinds of abstract services supported by the Fedora repository. The set of all *SDeps* represents a "registry" of all the concrete service bindings for the abstract service definitions supported by the Fedora repository. The set of all *CModels* represent a "registry" of the different user-defined types of data objects that exist in that Fedora repository.

At the end of the day, *FDOs* make references to *SDefs*, *SDeps* and *CModels* as the way of providing extended access points for *FDOs* (i.e., dynamic content disseminations.) This is done by adding special relationships between the objects that are stored in the RELS-EXT Datastreams of those objects. Figure 13 indicates the relationships that exist between the four object types. Data objects assert that they conform to a particular Content Model using the *hasModel* relationship. Content Model objects assert they provide the services included in an *SDef* using the *hasService* relationship. Service Deployment objects assert the services for which they provide binding information by using the *isDeploymentOf* relationship, as well as asserting the Content Models for which they provide service bindings using the *isContractorOf* relationship.

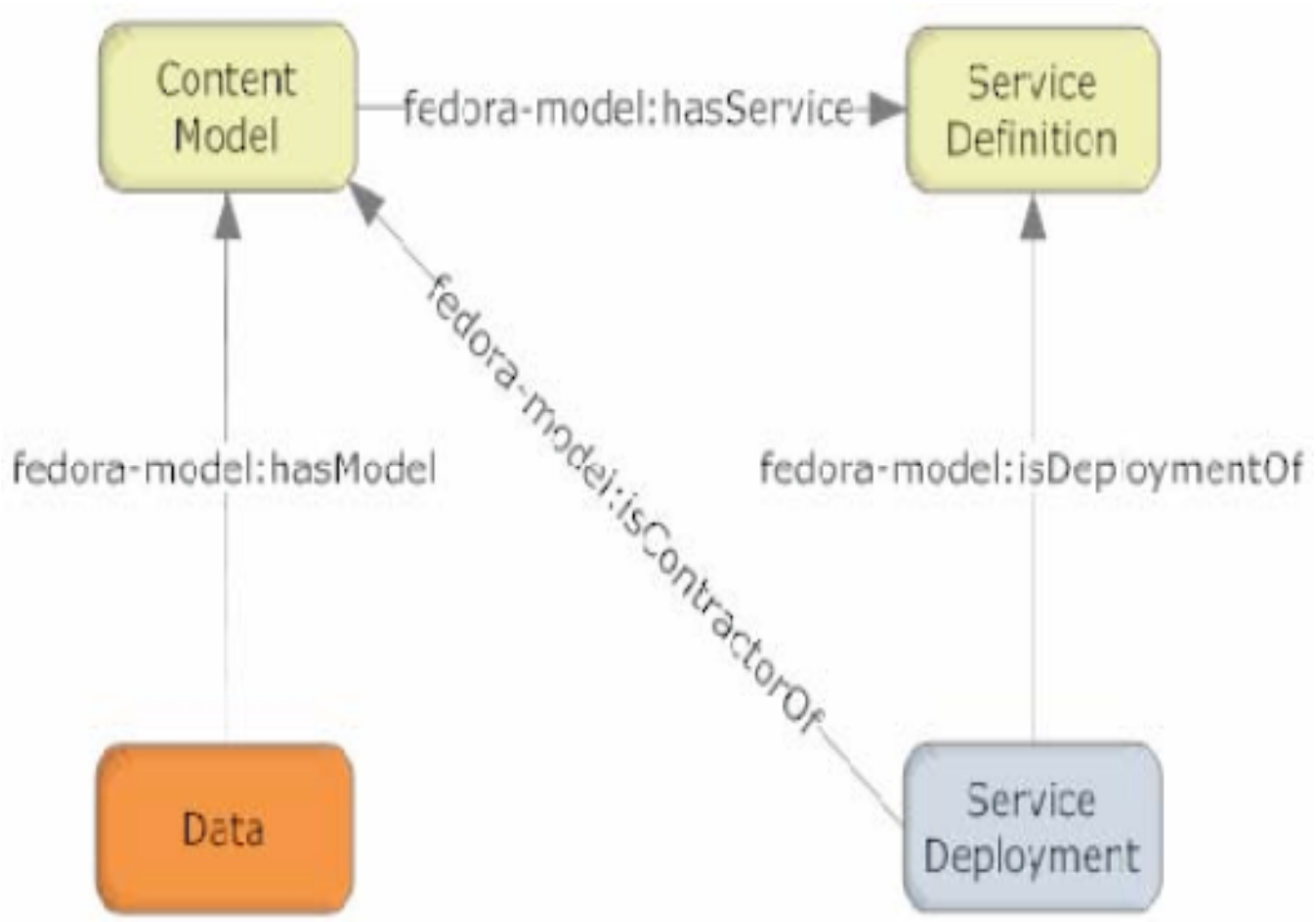


Figure 13 – Fundamental *Content Model Architecture* Relationships

Figure 14 illustrates the interactions among Fedora and Web services in response to an access request. As indicated, a client makes a request to the Fedora API (with a URL in this case.) The Fedora repository service then determines the content model that is associated with the *FDO* for which the request is being made. Once it knows the content model, the Fedora repository can discover what *SDefs* and *SDeps* are in play for this *FDO*. Once all of this information is gathered, the Fedora repository can construct a request to the appropriate web service to transform the Datastreams of the target *FDO* (`demo:2`). The Fedora repository service invokes a REST-based request to the web service via HTTP, sending along arguments to enable the web service to obtain the required Datastream inputs to fulfill the request. The Fedora repository mediates all invocations with the external web service. When it receives a response from the web service it streams it back to the original calling client. In this case, the response is a transformation based on the raw material of Datastream1 and Datastream2 in the *FDO*.

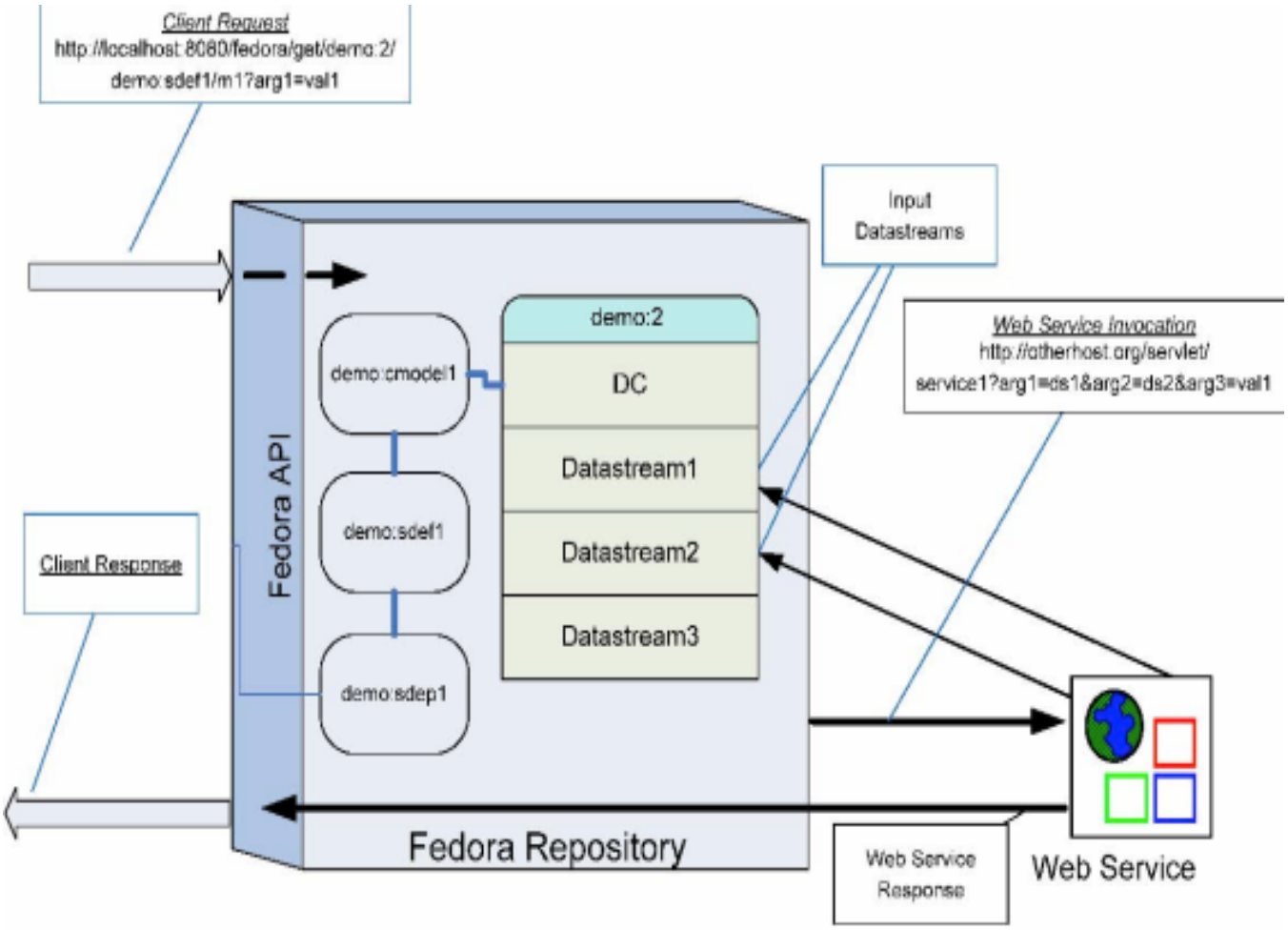


Figure 14 – Dynamic Dissemination Access

Example 3: Using *SDefs*, *SDeps* and *CModels*

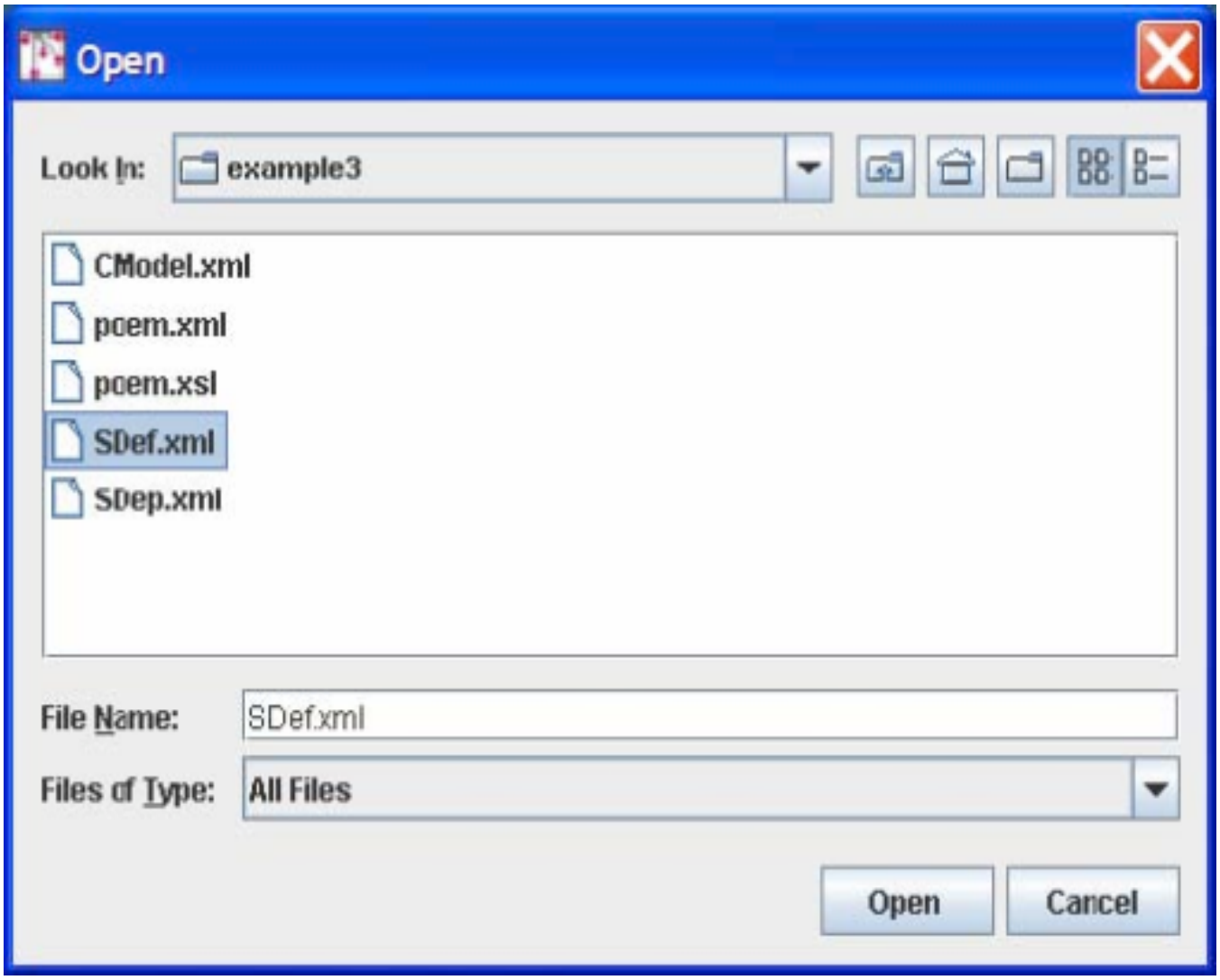
This example makes use of a *SDef*, *SDep* and *CModel* supplied with the Fedora tutorial. This will help you understand the basics of dynamic disseminations in Fedora under the Content Model Architecture, without writing a *SDef*, *SDep* or *CModel*. The next example describes how to do that more advanced task.

The web service used in the example performs an *XSL* transform using the well-known Saxon *XSLT* processor. This service requires two inputs, an *XML* source document and a *XSL* transform document. In this example, both of these *XML* documents are stored as managed content in a *Fedora Digital Object* . The *XML* source is data for a poem with tags for the structural elements of the poem (stanzas and lines). The *XSL* transform produces a *HTML* output of the poem that can be viewed in a browser. This example is borrowed from the web available source for Michael Kay's excellent *XSLT* book.

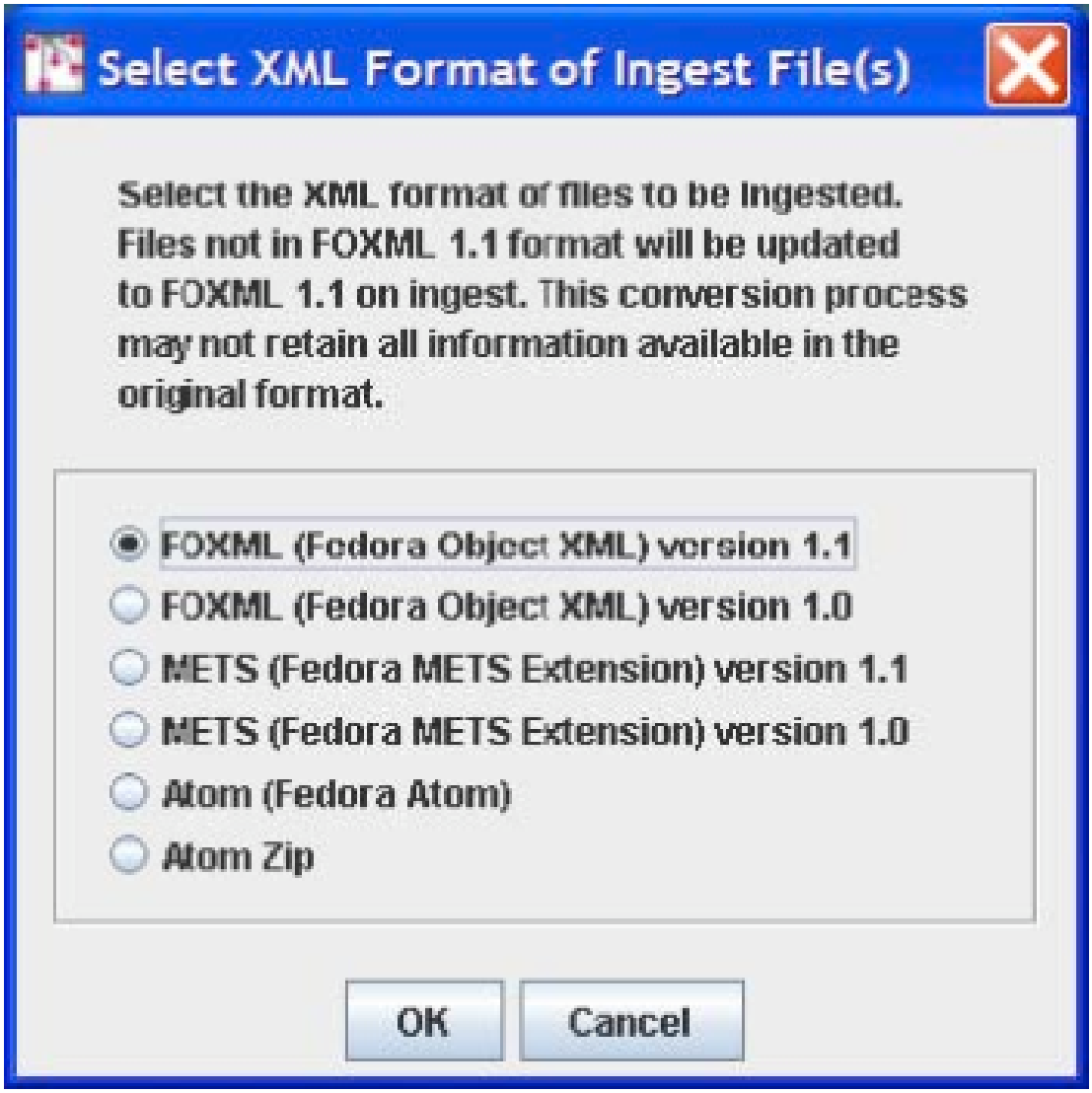
Ingesting Pre-defined *SDef*, *SDep* and *CModel* Objects

First we will ingest a sample *SDef* object into the repository.

Select File/Ingest/One Object/From File… in the Fedora Administrator. This will bring up a file selection dialog box as follows:



Browse the file system to select the ingest file for the *SDef* object whose file name is `FEDORA_HOME/userdocs/tutorials/2/example3/SDef.xml`. Since this ingest file is encoded as FOXML 1.1 select the FOXML 1.1 radio button as below:



This will create the *FDO* with *PID* `demo:ex3SDef` in your repository. This *SDef* defines one method getContent. This generic method name is intentional – one could imagine this one *SDef* being used as the basis for several *SDeps*, each of which produces "content" via a unique transformation of an underlying source. This is one of the advantages of Fedora – providing a common interface despite multiple underlying representations.

Follow the same procedure to ingest a sample *SDep* object into the repository. Select the file FEDORA_HOME/userdocs/tutorials/2/example3/SDep.xml. This will create the *FDO* with the *PID* `demo:ex3SDep`. This *SDep* represents a concrete implementation of the abstract service operations defined in the *SDef* `demo:ex3SDef`. The *SDep* object contains metadata that specifies the following:

- Service Contract: the *SDep* indicates the *PID* of the *SDef* that it is related to. This is like saying that the *SDep* provides and implementation of the *SDef*.
- Service binding metadata (i.e., in WSDL) : concrete binding for the getContent method that is defined. Specifically, the WSDL indicates that the getContent operation binding exists at the base URL of http://localhost:8080/service/saxon. Note that this service is hosted at the same host and port as the Fedora repository. As noted earlier, this is a local service that is packaged with Fedora.
- Data input profile that indicates that the *SDep*service operation getContent will take the following inputs at runtime:
  - "xsl" with *MIME* type text/xml.
  - "source" with *MIME* type text/xml.

Next follow the same procedure to ingest a sample *CModel* object into the repository. Select the file FEDORA_HOME/userdocs/tutorials/2/example3/CModel.xml. This will create the *FDO* with the *PID* `demo:ex3CModel`. This *CModel* describes the Datastreams that should be present in data objects that conform to this content model, it also has a RELS-EXT hasService relationship link to the *FDO* `demo:ex3SDef` ingested previously.

Creating a *Fedora Digital Object* with Appropriate *Datastreams*

Now you need to create the new *FDO* based on this *SDef*, *SDep* and *CModel*. To get started follow the same procedure as illustrated in Figure 3, this time entering demo:300 as the Datastream ID and Example 3 as the Label.

You now need to add the two Datastreams: the *XML* source document and the *XSL* transform document. Using the same method described in Example 1, select the Datastreams tab and:

- Add a Datastream with:

    - ID – source
    - Control Group – Managed Content
    - Mime type – text/xml
    - Label – Poem XML Source
    - Import location: FEDORA_HOME/userdocs/tutorials/2/example3/poem.xml
- Add a Datastream with:
    - ID – xsl
    - Control Group – Managed Content
    - Mime type – text/xml
    - Label - Poem XSL Transform
    - Import location: FEDORA_HOME/userdocs/tutorials/2/example3/poem.xsl

Linking the *Fedora Digital Object* to the *Content Model*

In Fedora Administrator, select the Datastreams tab from the *FDO* window, and then select the New RELS-EXT… tab. The resulting dialog will now allow you to create the necessary RELS-EXT relationship to allow dynamic dissemination to work. Follow these steps:

1. Select the Edit button, then the Add… button to create a new relationship.
2. In the Enter Relationship dialog that appears, in the Predicate: drop-down dialog, select the entry fedora-model:hasModel and in the Object: text entry box, enter the string info:fedora/demo:ex3CModel, and then press the OK button.
3. hould then see the newly created relationship in the table at the bottom of the New RELS-EXT… window. Press the Save Datastream button to save this newly created Datastream.

The resulting Object window should look like that illustrated in Figure 15.

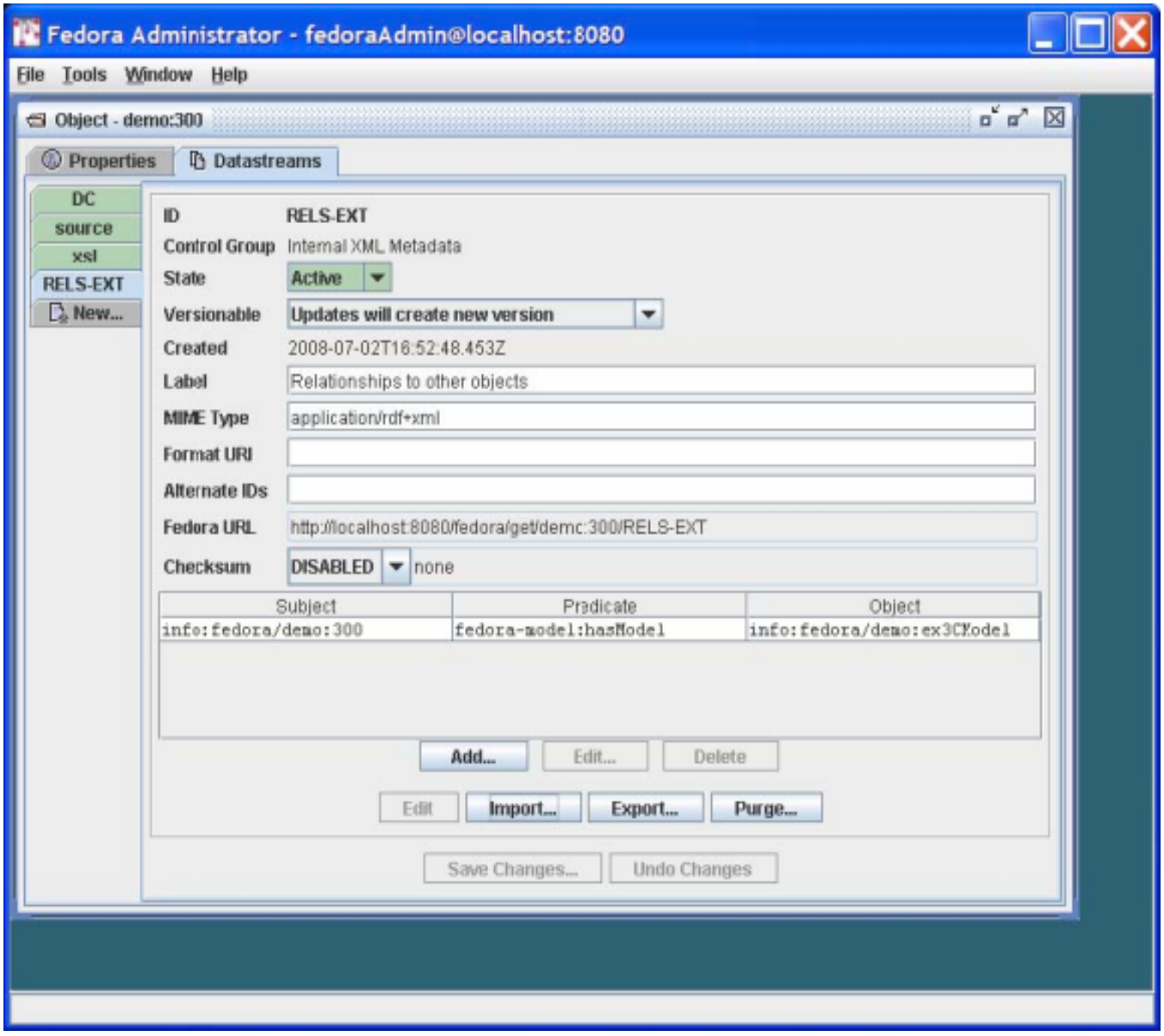


Figure 15 – Example 3 Linking a *Fedora Digital Object* to a *Content Model*

You're done! Figure 16 illustrates the role of this *FDO* and dissemination service in response to a client request. You can go to the *FDO* header page at http://localhost:8080/fedora/get/demo:300 and select the View Dissemination Index link. Your newly added dynamic dissemination should now appear, alongside the primitive behaviors for the object. To see the results of this dynamic dissemination, you can either select the Run button for getContent in the Method Index display or simply enter the URL `http://localhost:8080/fedora/get/demo:300/demo:ex3SDef/getContent` directly.

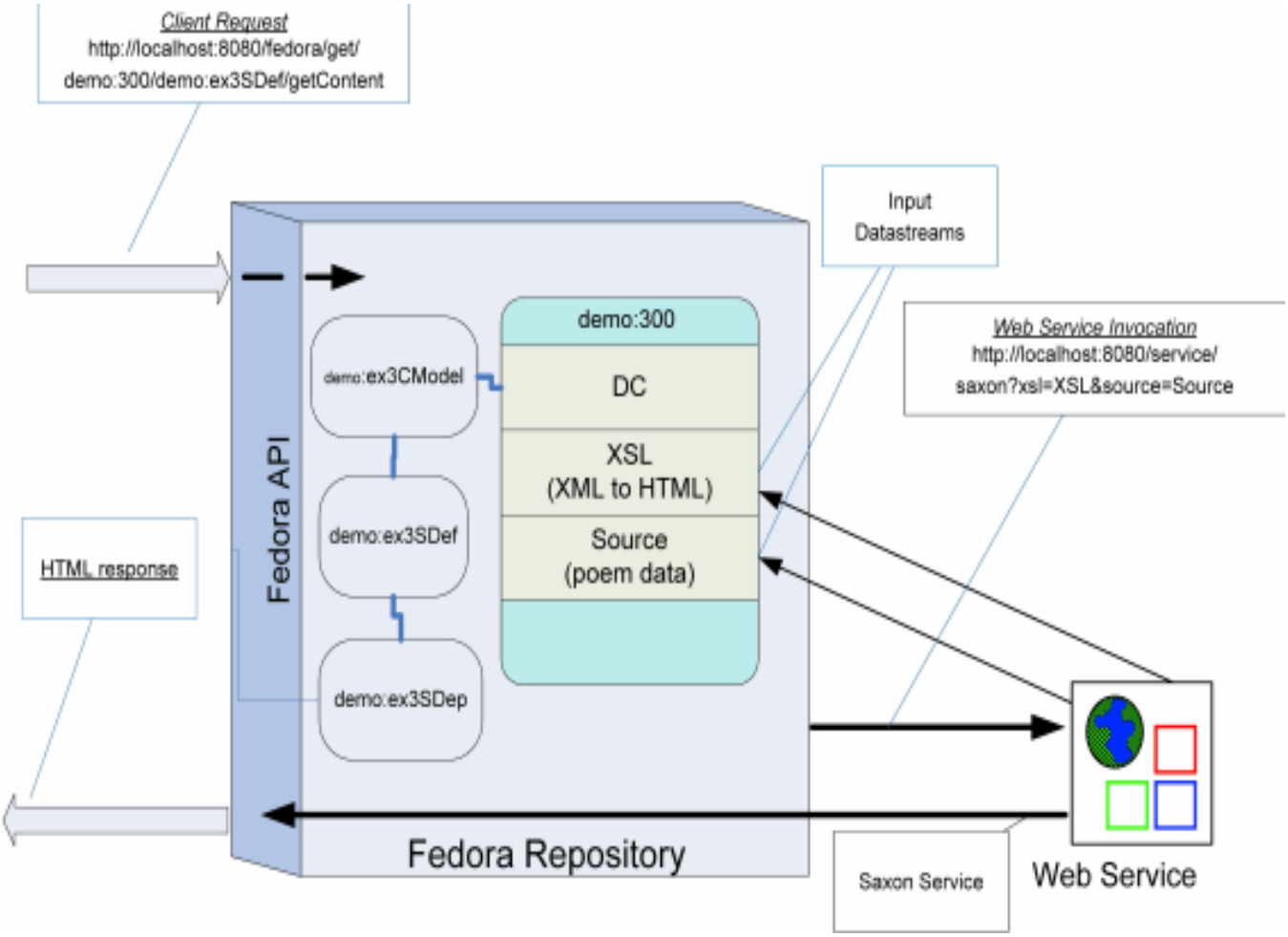


Figure 16 – Example 3 dissemination via the *Content Model Architecture*

Example 4 – Modifying Example 3 Using a Redirect *Datastream*

Example 3 packages the *XSL* transform Datastream in the same *FDO* as the source *XML* Datastream. However, in many cases you will have *XSL* transform code that you want to share across several *XML* sources. This section modifies Example 3 to enable this sharing.

This is done by packaging the *XSL* transform code in a *FDO* of its own. Then every *FDO* that needs to make use of the *XSL* transform code can use the Fedora REST URL to access that Datastream. This is done by defining a redirect Datastream using the REST URL as the redirect target. Then, the same disseminator design used in Example 3 can be reused. This is known as *dissemination chaining*, whereby the dissemination of one *FDO* is used by another.

The steps to do this are quite simple and use techniques introduced thus far:

- Create a new *FDO* (the *XSL FDO*) assigning the *PID* `demo:400`. Create one Datastream in addition to the DC with ID XSL. As before, this Datastream should be configured as:
    - ID – xsl
    - Control Group – Managed Content
    - Mime type – text/xml
    - Label - Poem XSL Transform
    - Import location: FEDORA_HOME/userdocs/tutorials/2/example3/poem.xsl
- Create another *FDO* (the "dissemination service" *FDO*) assigning the *PID* `demo:500`.
- Create two new Datastreams
    - One configured as follows (the same as the Source Datastream in Example 3):
        1. ID – source
        2. Control Group – Managed Content
        3. MIME type – text/xml
        4. Label - Poem XML Source
        5. Import location: FEDORA_HOME/userdocs/tutorials/2/example3/poem.xml
    - Now create the Datastream that will redirect to the *XSL*in demo:400 as follows:

1. ID – xsl
2. Control Group – Redirect
3. Mime Type – text/xml
4. Label - Poem XSL Transform
5. location: http://localhost:8080/fedora/get/demo:400/XSL
6. • On the New RELS-EXT… tab add the same hasContentModel relationship to demo:ex3CModel as you did in example 3.

You're done! The `demo:500` *FDO* should now behave exactly the same as the `demo:300` *FDO* in Example 3. Figure 17 refines Figure 16 (with some labeling removed for clarity) with the new redirect configuration.

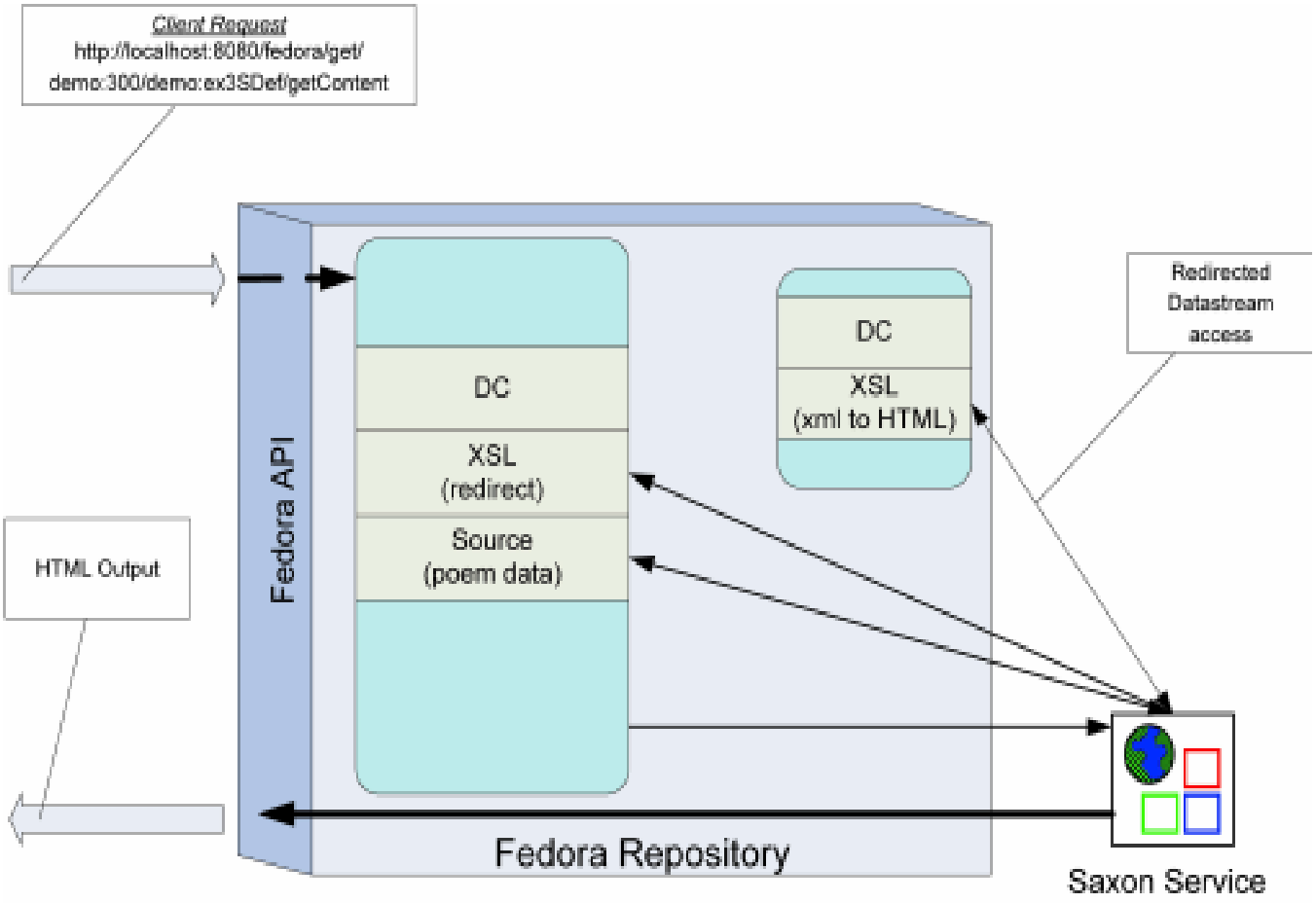


Figure 17 - Dissemination with Redirect *Datastream*

What's next?

You should now understand the basic mechanisms through which *SDefs*, *SDeps* and *CModels* interact with Data objects to provide a richer dynamic view of the data stored in those objects. The next tutorial (Tutorial 3 – Not yet available) steps you through the process of using the admin client to create a *SDef* , a *SDep*, and a *CModel* from scratch and a *Data Object* that will function with the control objects to provide customized services similar to those described in the last example of this tutorial. To explore the other features of Fedora, refer to the Fedora Repository Documentation. You can also join the Fedora-users mailing list to ask questions and learn from the experience of other Fedora users.