# Versioning

Unable to render {include} The included page could not be found.

## Introduction

As of v1.2, Fedora for the first time included content versioning functionality. Content versioning is one of the preservation-oriented features of Fedora. Using this new capability, Fedora repositories can maintain a record of how digital objects have changed over time. This is achieved by the Fedora repository storing former versions of content, and by creating audit trail records about changes. Also, with the advent of content versioning, the Fedora Access interfaces now support date-time stamped requests, so that a client can "go back in time" and see a digital object as it looked in the past.

As of v2.2, the versioning capabilities of Fedora have been enhanced. Although the default operation for digital object in a Fedora Repository is still that any modification of a Datastream will create and store a new version of that Datastream, it is now possible to have a greater degree of control over the versioning capability. Now whenever a Datastream is created (either through the createDatastream API function, or as a part of ingesting an entire digital object) the user can specify whether that specific Datastream is to be versioned or not.

Additionally, there are now API functions that allow the user greater control of which modifications of Datastreams stored for a digital object are important enough to merit maintaining the version in perpetuity, and which are less important administrative changes which need not be preserved.

## Preservation

The ability to version content of Fedora data objects provides repositories with the ability to preserve not only the data for any given data object over time, but also to preserve the look and feel of the dissemination of that data object, since the original content is linked to the original delivery mechanisms used to disseminate that content.

## How It Works by Default

In the more versatile versioning system now provided by Fedora, the default operation is that any modifications made to a Datastream through the Fedora management interface (API-M) will automatically result in a new version of that Datastream being created by Fedora. Fedora will not create a new version of the whole digital object, instead it will version these specific components within the digital object container. This has the benefit of the digital object PID remaining constant, and not having to keep track of multiple distinct versions of a digital object.

The Fedora repository typically will maintain all versions of all Datastreams, thereby creating a history of how objects change over time. Additionally, Fedora maintains an audit trail record of the nature of the object change events (e.g., who, what, when, why).

While the components of data objects are versioned in Fedora v2.0 through 2.2, versioning of Service Definitions and Service Deployments) is in scope for future releases of the software.

## Disabling Versioning

As of v2.2, the Fedora versioning system allows a user to selectively *disable* versioning for a given Datastream of a digital object (via the new API function setDatastreamVersionable). Subsequently, any modifications made to that specific Datastream will *replace* the most recent stored version with the result of the modification. This replacement only applies to the most recent version of that Datastream, any versions that were in existence before versioning was disabled will remain unchanged. Thereafter if the user decides that new modifications ought to cause new versions to be stored, versioning can be turned back on (again through the API function set DatastreamVersionable).

## Purging Old Versions

To provide more control over what versions of a particular Datastream are to be kept, the API function purgeDatastream has been made more versatile. Previously purgeDatastream could only be used to delete the oldest version (or versions) of a Datastream. If, for instance a Datastream had six different versions that had been created over its lifetime (numbered 1 through 6 from oldest to most recent) and a user wanted to get rid of the fourth stored version, the only action the user could take previously would be to delete versions 1 through 4.

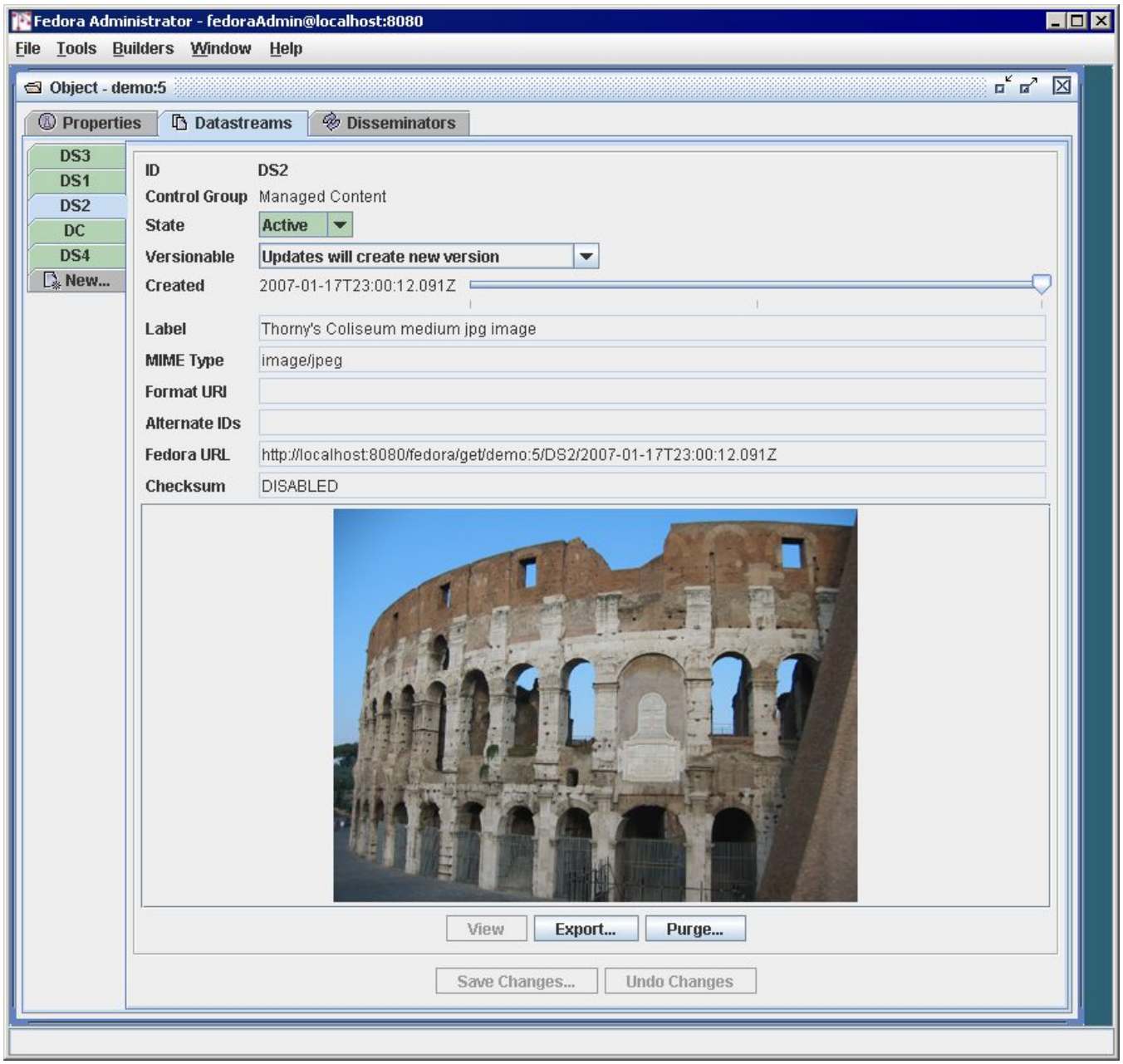
Now rather than specifying single date, and deleting all versions older than that date, the purgeDatastream function allows a start date and end date to be specified which will allow any single previous version (or any contiguous range of versions) to be deleted.

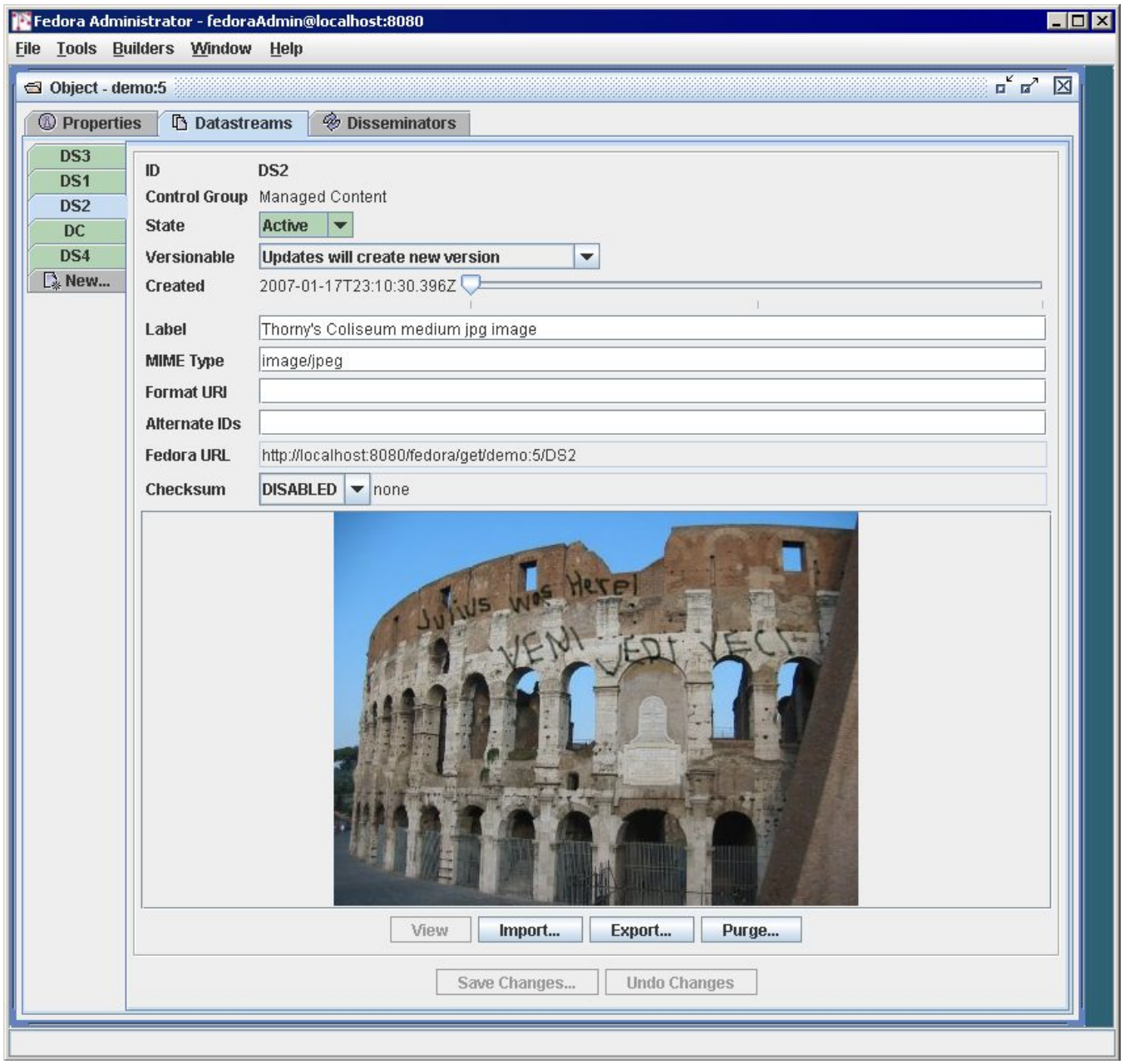
## Fedora Administrator

Any component of an object that may be modified is shown in Fedora Administrator as either an editable text box or by means of an Edit button on the active pane. Any modification to a data object component results in a new version of that component being created and stored in the XML of the data object.

Previous versions of an edited Datastream may be viewed, but cannot be edited. A slider bar is provided to indicate that previous versions of a Datastream exist. Versions are represented on this timeline by small vertical lines. Clicking on the bar will move the slider to the points on the timeline. Additionally the slider may be dragged along the timeline to show all versions.

The following screen shots show a versioned image Datastream. Note that a dropdown box is present on the Datastream pane indicating whether versioning is enabled for that particular Datastream (ie. whether Updates will create new version or Updates will replace the most recent version):


Fedora Administrator - fedoraAdmin@localhost:8080
File Tools Builders Window Help
Object - demo:5
Properties
Datastreams
Disseminators
DS3
DS1
DS2
DC
DS4
New...
ID DS2
Control Group Managed Content
State Active
Versionable Updates will create new version
Created 2007-01-17T23:00:12.091Z
Label Thorny's Coliseum medium jpg image
MIME Type image/jpeg
Format URI
Alternate IDs
Fedora URL http://localhost:8080/fedora/get/demo:5/DS2/2007-01-17T23:00:12.091Z
Checksum DISABLED
View
Export...
Purge...
Save Changes...
Undo Changes




## Search and Retrieval in Fedora Administrator

Retrieval of a modified data object requires using the Advanced Search tab on the Search Repository dialog and entering a specific date/time stamp as a search criteria.

All components are given a created date (cDate) value at ingest. Whenever the component is modified, the modified date (mDate) value is updated. By searching on the modified date, it is possible to retrieve a result set of modified data objects. Wild card values are allowed in the date/time search criteria. For example:

```
mDate<=2003-08-28T*
and
mDate<=2003-08-28T23:59:59
```

are both valid search criteria.

In the case of Dublin Core Metadata Datastreams, the modified date is stored in the dcmDate parameter. To retrieve data objects with modified Dublin Core Metadata Datastreams, the user must search using this parameter name.

## API-A-Lite Web Interface

Retrieval of specific versions of objects via API-A-Lite also depends on entering date/time parameter values. This section gives the syntax for the getDissemination and getObjectSyntax methods.

If no date/time value is used in the URL, the most current version of a data object is always returned as the result of a search.

**getDissemination syntax:** http://hostname:port/fedora/get/PID/sDefPID/methodName*[/dateTime][?parmArray]*

This syntax requests a dissemination of the specified object using the specified method of the associated behavior definition object. The result is returned as a MIME-typed stream.

- hostname - required hostname of the Fedora server.
- port - required port number on which the Fedora server is running. fedora - required name of the Fedora access service.
- fedora - a required parameter specifying the Fedora servlet path.
- get - a required parameter specifying the Fedora servlet path.
- PID - required persistent identifier of the digital object.
- sDefPID - required persistent identifier of the service definition object to which the digital object subscribes.
- methodName - required name of the method to be executed.
- **dateTime** - value indicating dissemination of a version of the digital object at the specified point in time. Proper syntax is YYYY-MM-DDTHH:MM:SS where HH is 24-hour clock.
- **parmArray** - optional array of method parameters consisting of name/value pairs in the form parm1=value1&parm2=value2...

For example: http://localhost:8080/fedora/get/ronda:2/demo:1/getItem/2003-08-28T00:01:01

**getObjectProfile syntax:** http://hostname:port/fedora/get/PID*[/dateTime][?xml=BOOLEAN]*

This syntax requests an object profile for the specified digital object. The XML parameter determines the type of output returned. If the parameter is omitted or has a value of "false," a MIME-typed stream consisting of an HTML table is returned providing a browser-savvy means of viewing the object profile. If the value specified is "true," then a MIME-typed stream consisting of XML is returned.

- hostname - required hostname of the Fedora server.
- port - required port number on which the Fedora server is running. fedora - required name of the Fedora access service.
- fedora - a required parameter specifying the Fedora servlet path.
- get - a required parameter specifying the Fedora servlet path.
- PID - required persistent identifier of the digital object.
- **dateTime** - value indicating dissemination of a version of the digital object at the specified point in time. Proper syntax is YYYY-MM-DDTHH:MM:SS where HH is 24-hour clock.
- **xml** - an optional parameter indicating the requested output format. A value of "true" indicates a return type of text/xml; the absence of the xml parameter or a value of "false" indicates format is to be text/html.

For example: http://localhost:8080/fedora/get/ronda:2/2003-08-28T00:01:01

Unable to render {include} The included page could not be found.