REST API

- Introduction
- API-A Methods
  - describeRepository
  - findObjects
  - getDatastreamDissemination
  - getDissemination
  - getObjectHistory
  - getObjectProfile
  - listDatastreams
  - listMethods
  - resumeFindObjects
- API-M Methods
  - addDatastream
  - addRelationship
  - compareDatastreamChecksum
  - export
  - getDatastream
  - getDatastreamHistory
  - getDatastreams
  - getNextPID
  - getObjectXML
  - getRelationships
  - ingest
  - modifyDatastream
  - modifyObject
  - purgeDatastream
  - purgeObject
  - purgeRelationship
  - setDatastreamState
  - setVersionable
  - Validate
- Utility Methods
  - Upload
- WADL

Introduction

The Fedora REST API exposes a subset of the Fedora Access and Management APIs as a RESTful (Representational State Transfer) Web Service. For release 3.2, the Fedora REST API has been upgraded to Beta status. With this change the REST API is no longer optional, it is enabled by default as are all of the other Fedora APIs. The primary reasons behind Beta status are the need for more robust testing, as well as the understanding that in a future release the REST API will likely subsume API-A-Lite and API-M-Lite, thus changing the API somewhat.

For examples of how to use the REST API programmatically, please refer to the TestRESTAPI test class (http://fedora-commons.svn.sourceforge.net/viewvc/*checkout*/fedora-commons/fedora/tags/release-3.0/src/test/junit/fedora/test/api/TestRESTAPI.java).

Ensure DC, RELS-EXT and RELS-INT are versionable if using Managed Content

Due to an outstanding bug FCREPO-849, if you use Managed Content for DC, RELS-EXT or RELS-INT then please make sure these datastreams are versionable (the default setting for versionable is "true", so if you haven’t specified this datastream property then you are safe). Ensure that you don't inadvertently set this property to “false” for these datastreams when using the API methods.

2xx Responses only please

The HTTP Response portion of each method description listed below indicates the response on success. Unsuccessful calls will produce non-200 response codes appropriate to the error case. If, however, your client software has difficulty processing non-200 responses (such as is the case with Adobe’s Flash Player) adding the query parameter ‘flash=true’ to any method will ensure that all responses are in the 200 range. In the event of an error, the response code will be set to 200 and the response body will include the error message followed by "::ERROR".

POST Replacement

If the client with which you are working does not support use of the PUT and/or DELETE HTTP methods but does allow you to set headers on the HTTP request, you can use POST replacement to make PUT and DELETE calls. To do this, simply set the X-HTTP-Method-Override request header to the correct method value (PUT or DELETE) and perform a POST request. Your request will be handled by the REST API as if it were a PUT or DELETE.
**API-A Methods**

**describeRepository**
Not implemented

**findObjects**

**URL Syntax**

```
/objects ? [terms | query] [maxResults] [resultFormat] [pid] [label] [state] [ownerId] [cDate] [mDate] [dcmDate] [title] [creator] [subject] [description] [publisher] [contributor] [date] [type] [format] [identifier] [source] [language] [relation] [coverage] [rights]
```

**HTTP Method**

GET

**HTTP Response**

200

**Parameters**

**Removal of .xml shortcut**

For release 3.3 the .xml shortcut has entirely been removed from the REST API due to functional inconsistencies (see [here](#) for more details. If your client uses this shortcut please change it to use the format parameter (/?format=xml).

**URL-Encoding**

The REST API requires that parameters - including path parameters - are URL-encoded. Particularly this is important if you have any PIDs that use escaped-octets in the PID name. In this case the "%" character should be URL-encoded as "%25", eg a PID "changeme:1234%2F56" should be URL-encoded as "changeme:1234%252F56". The ":" PID namespace separator character does not require URL-encoding as it has no special meaning when used in the path component of HTTP URIs; however some software library URL-encoding methods will URL-encode this to %3A - that's not a problem, both ":" and "%3A" will be accepted by the REST API.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>terms</td>
<td>a phrase represented as a sequence of characters (including the ? and * wildcards) for the search. If this sequence is found in any of the fields for an object, the object is considered a match. Do NOT use this parameter in combination with the “query” parameter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>query</td>
<td>a sequence of space-separated conditions. A condition consists of a metadata element name followed directly by an operator, followed directly by a value. Valid element names are (pid, label, state, ownerId, cDate, mDate, dcmDate, title, creator, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage, rights). Valid operators are: contains (.), equals (=), greater than (&gt;), less than (&lt;), greater than or equals (&gt;=), less than or equals (&lt;=). The contains (.) operator may be used in combination with the ? and * wildcards to query for simple string patterns. Space-separators should be encoded in the URL as %20. Operators must be encoded when used in the URL syntax as follows: the (=) operator must be encoded as %3D, the (&gt;) operator as %3E, the (&lt;=) operator as %3C%3D, the (&gt;=) operator as %3E%3D, and the (~) operator as %7E. Values may be any string. If the string contains a space, the value should begin and end with a single quote character ('). If all conditions are met for an object, the object is considered a match. Do NOT use this parameter in combination with the &quot;terms&quot; parameter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxResults</td>
<td>the maximum number of results that the server should provide at once. If this is unspecified, the server will default to a small value.</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>resultFormat</td>
<td>the preferred output format</td>
<td>html</td>
<td>xml, html</td>
</tr>
<tr>
<td>pid</td>
<td>if true, the Fedora persistent identifier (PID) element of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>label</td>
<td>if true, the Fedora object label element of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>state</td>
<td>if true, the Fedora object state element of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>ownerId</td>
<td>if true, each matching objects' owner id will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>cDate</td>
<td>if true, the Fedora create date element of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>mDate</td>
<td>if true, the Fedora modified date of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>dcmDate</td>
<td>if true, the Dublin Core modified date element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>title</td>
<td>if true, the Dublin Core title element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>creator</td>
<td>if true, the Dublin Core creator element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>subject</td>
<td>if true, the Dublin Core subject element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>description</td>
<td>if true, the Dublin Core description element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>publisher</td>
<td>if true, the Dublin Core publisher element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>contributor</td>
<td>if true, the Dublin Core contributor element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>date</td>
<td>if true, the Dublin Core date element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>type</td>
<td>if true, the Dublin Core type element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>format</td>
<td>if true, the Dublin Core format element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>identifier</td>
<td>if true, the Dublin Core identifier element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>source</td>
<td>if true, the Dublin Core source element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>language</td>
<td>if true, the Dublin Core language element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>relation</td>
<td>if true, the Dublin Core relation element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>coverage</td>
<td>if true, the Dublin Core coverage element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>rights</td>
<td>if true, the Dublin Core rights element(s) of matching objects will be included in the response</td>
<td>false</td>
<td>true, false</td>
</tr>
</tbody>
</table>

**Examples**

/objects?terms=demo&pid=true&subject=true&label=true&resultFormat=xml
/objects?query=title%7Erome%20creator%7Estaples&pid=true&title=true&creator=true
/objects?query=pid%7E*1&maxResults=50&format=true&pid=true&title=true

**getDataStreamDissemination**
URL Syntax
/objects/{pid}/datastreams/{dsID}/content ? [asOfDateTime] [download]

HTTP Method
GET

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pid)</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(dsID)</td>
<td>datastream identifier</td>
<td></td>
<td></td>
</tr>
<tr>
<td>asOfDateTime</td>
<td>indicates that the result should be relative to the digital object as it existed at the given date and time</td>
<td></td>
<td>yyyy-MM-dd or yyyy-MM-ddTHH:mm:ssZ</td>
</tr>
<tr>
<td>download</td>
<td>If true, a content-disposition header value &quot;attachment&quot; will be included in the response, prompting the user to save the datastream as a file. A content-disposition header value &quot;inline&quot; will be used otherwise. The filename used in the header is generated by examining in order: RELS-INT for the relationship fedora-model:downloadFilename, the datastream label, and the datastream ID. The file extension (apart from where the filename is specified in RELS-INT) is determined from the MIMETYPE. The order in which these filename sources are searched, and whether or not to generate an extension from the MIMETYPE, is configured in fedora.fcfg. The file used to map between MIMETYPEs and extensions is mime-to-extensions.xml located in the server config directory.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples
/objects/demo:29/datastreams/DC/content
/objects/demo:29/datastreams/DC/content?asOfDateTime=2008-01-01

getDissemination

URL Syntax
/objects/{pid}/methods/{sdefPid}/{method} ? [method parameters]

HTTP Method
GET

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pid)</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(sdefPid)</td>
<td>persistent identifier of the sDef defining the methods</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(method)</td>
<td>method to invoke</td>
<td></td>
<td></td>
</tr>
<tr>
<td>method parameters</td>
<td>any parameters required by the method</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples
/objects/demo:29/methods/demo:27/resizeImage?width=100
/objects/demo:SmileyEarring/methods/demo:DualResolution/fullSize

ggetObjectHistory
URL Syntax
/objects/{pid}/versions ? [format]

HTTP Method
GET

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pid)</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>the preferred output format</td>
<td>html</td>
<td>xml, html</td>
</tr>
</tbody>
</table>

Examples
/objects/demo:29/versions
/objects/demo:29/versions?format=xml

getObjectProfile

URL Syntax
/objects/{pid} ? [format] [asOfDateTime]

HTTP Method
GET

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pid)</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>the preferred output format</td>
<td>html</td>
<td>xml, html</td>
</tr>
<tr>
<td>asOfDateTime</td>
<td>indicates that the result should be relative to the digital object as it existed on the given date</td>
<td></td>
<td>yyyy-MM-dd or yyyy-MM-ddTHH:mm:ssZ</td>
</tr>
</tbody>
</table>

Examples
/objects/demo:29
/objects/demo:29?format=xml
/objects/demo:29?asOfDateTime=2008-01-01

listDatastreams
URL Syntax
/objects/{pid}/datastreams ? [format] [asOfDateTime]

HTTP Method
GET

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>{pid}</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>the preferred output format</td>
<td>html</td>
<td>xml, html</td>
</tr>
<tr>
<td>asOfDateTime</td>
<td>indicates that the result should be relative to the digital object as it existed on the given date</td>
<td>yyyy-MM-dd or yyyy-MM-ddTHH:mm:ssZ</td>
<td></td>
</tr>
</tbody>
</table>

Examples
/objects/demo:35/datastreams
/objects/demo:35/datastreams?format=xml&asOfDateTime=2008-01-01T05:15:00Z

listMethods

URL Syntax
1. /objects/{pid}/methods ? [format] [asOfDateTime]
2. /objects/{pid}/methods/{sdefPid} ? [format] [asOfDateTime]

HTTP Method
GET

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>{pid}</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{sdefPid}</td>
<td>persistent identifier of the SDef defining the methods</td>
<td></td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>the preferred output format</td>
<td>html</td>
<td>xml, html</td>
</tr>
<tr>
<td>asOfDateTime</td>
<td>indicates that the result should be relative to the digital object as it existed on the given date</td>
<td>yyyy-MM-dd or yyyy-MM-ddTHH:mm:ssZ</td>
<td></td>
</tr>
</tbody>
</table>

Examples
/objects/demo:29/methods
/objects/demo:29/methods?format=xml&asOfDateTime=2008-01-01T05:15:00Z
/objects/demo:29/methods/demo:27
/objects/demo:29/methods/demo:27?format=xml&asOfDateTime=2008-01-01T05:15:00Z

resumeFindObject
URL Syntax
/objects [sessionToken] [all findObjects options]

HTTP Method
GET

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>sessionToken</td>
<td>the identifier of the session to which the search results are being returned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>all findObjects options</td>
<td>all of the same options are available for resumeFindObjects as for findObjects</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples
/objects?terms=*&format=xml&pid=true&subject=true&label=true&sessionToken=xyz

API-M Methods

addDatastream
URL Syntax
/objects/(pid)/datastreams/(dsID) ? [controlGroup] [dsLocation] [altIDs] [dsLabel] [versionable] [dsState] [formatURI] [checksumType] [checksum] [mimeType] [logMessage]

HTTP Method
POST

HTTP Response
201

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pid)</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(dsID)</td>
<td>datastream identifier</td>
<td></td>
<td></td>
</tr>
<tr>
<td>controlGroup</td>
<td>one of “X”, “M”, “R”, or “E” (Inline <em>X</em>ML, <em>M</em>anaged Content, <em>R</em>edirect, or <em>E</em>xternal Referenced)</td>
<td>X</td>
<td>X, M, R, E</td>
</tr>
<tr>
<td>dsLocation</td>
<td>location of managed or external datastream content</td>
<td></td>
<td></td>
</tr>
<tr>
<td>altIDs</td>
<td>alternate identifiers for the datastream</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dsLabel</td>
<td>the label for the datastream</td>
<td></td>
<td></td>
</tr>
<tr>
<td>versionable</td>
<td>enable versioning of the datastream</td>
<td>true</td>
<td>true, false</td>
</tr>
<tr>
<td>formatURI</td>
<td>the format URI of the datastream</td>
<td></td>
<td></td>
</tr>
<tr>
<td>checksumType</td>
<td>the algorithm used to compute the checksum</td>
<td>DEFAULT</td>
<td>DEFAULT, DISABLED, MD5, SHA-1, SHA-256, SHA-385, SHA-512</td>
</tr>
<tr>
<td>checksum</td>
<td>the value of the checksum represented as a hexadecimal string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mimeType</td>
<td>the MIME type of the content being added, this overrides the Content-Type request header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>logMessage</td>
<td>a message describing the activity being performed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>multipart file as request content</td>
<td>datastream file (for Managed datastreams)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples
POST: /objects/demo:29/datastreams/NEWDS?controlGroup=X&dsLabel=New (with Multipart file)


addRelationship
URL Syntax
/objects/{pid}/relationships/new ? [subject] [predicate] [object] [isLiteral] [datatype]

HTTP Method
POST

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pid)</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>subject</td>
<td>subject of the relationship. Either a URI for the object or one of its datastreams</td>
<td>URI of this object</td>
<td></td>
</tr>
<tr>
<td>predicate</td>
<td>predicate of the relationship</td>
<td></td>
<td></td>
</tr>
<tr>
<td>object</td>
<td>object of the relationship</td>
<td></td>
<td></td>
</tr>
<tr>
<td>isLiteral</td>
<td>true if the object of the relationship is a literal, false if it is a URI</td>
<td>true, false</td>
<td></td>
</tr>
<tr>
<td>datatype</td>
<td>if the object is a literal, the datatype of the literal (optional)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples
POST /objects/demo:29/relationships/new?subject=info%3afedora%2fdemo%3a29%2fDC&predicate=http%3a%2f%2fwww.example.org%2frels%2fname&object=dublin%20core&isLiteral=true

compareDatastreamChecksum

See #getDatastream

export

URL Syntax
/objects/{pid}/export ? [format] [context] [encoding]

HTTP Method
GET

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pid)</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>the XML format to export</td>
<td>info:fedora /fedora-system: FOXML-1.1</td>
<td></td>
</tr>
<tr>
<td>context</td>
<td>the export context, which determines how datastream URLs and content are represented</td>
<td>public</td>
<td>public, migrate, archive</td>
</tr>
<tr>
<td>encoding</td>
<td>the preferred encoding of the exported XML</td>
<td>UTF-8</td>
<td></td>
</tr>
</tbody>
</table>

Examples
/objects/demo:29/export
/objects/demo:29/export?context=migrate
**getDataSetream**

**URL Syntax**
/objects/{pid}/datastreams/{dsID} ? [asOfDateTime] [format] [validateChecksum]

**HTTP Method**
GET

**HTTP Response**
200

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pid)</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(dsID)</td>
<td>datastream identifier</td>
<td></td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>the preferred output format</td>
<td>html</td>
<td>xml, html</td>
</tr>
<tr>
<td>asOfDateTime</td>
<td>indicates that the result should be relative to the digital object as it existed on the given date</td>
<td>yyyy-MM-dd or yyyy-MM-ddTHH:mm:ssZ</td>
<td></td>
</tr>
<tr>
<td>validateChecksum</td>
<td>verifies that the Datastream content has not changed since the checksum was initially computed. If asOfDateTime is null, Fedora will use the most recent version.</td>
<td>false</td>
<td>true, false</td>
</tr>
</tbody>
</table>

**Examples**

/objects/demo:29/datastreams/DC

/objects/demo:29/datastreams/DC?format=xml

/objects/demo:29/datastreams/DC?format=xml&validateChecksum=true

**getDataSetreamHistory**

**URL Syntax**
/objects/{pid}/datastreams/{dsid}/history ? [format]

**HTTP Method**
GET

**HTTP Response**
200

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>format</td>
<td>the preferred output format</td>
<td>html</td>
<td>xml, html</td>
</tr>
</tbody>
</table>

**Examples**

GET: /objects/changeme:1/datastreams/DC/history

GET: /objects/changeme:1/datastreams/DC/history?format=xml

**getDataSetreams**

Not implemented

**getNextPID**
URL Syntax
/objects/nextPID ? [numPIDs] [namespace] [format]

HTTP Method
POST

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>numPIDs</td>
<td>the number of pids to retrieve</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>namespace</td>
<td>the namespace of the requested pid(s)</td>
<td>the default namespace of the repository</td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>the preferred output format</td>
<td>html</td>
<td>xml, html</td>
</tr>
</tbody>
</table>

Examples
POST: /objects/nextPID
POST: /objects/nextPID?numPIDs=5&namespace=test&format=xml

getObjectXML

URL Syntax
/objects/{pid}/objectXML

HTTP Method
GET

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>{pid}</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples
/objects/demo:29/objectXML

getRelationships
**URL Syntax**

/objects/{pid}/relationships ? [subject] [predicate] [format]

**HTTP Method**

GET

**HTTP Response**

200

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>{pid}</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>subject</td>
<td>subject of the relationship(s). Either a URI for the object or one of its datastreams</td>
<td>URI of this object</td>
<td></td>
</tr>
<tr>
<td>predicate</td>
<td>predicate of the relationship(s), if missing returns all predicates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>format of the response</td>
<td>rdf/xml</td>
<td>xml (returns rdf/xml), rdf/xml, n-triples, turtle, sparql</td>
</tr>
</tbody>
</table>

**Examples**

/objects/demo:29/relationships

/objects/demo:29/relationships?subject=info%3afedora%2fdemo%3a29%2fDC

**ingest**
**URL Syntax**

`/objects/ [{pid}| new] ? [label] [format] [encoding] [namespace] [ownerId] [logMessage] [ignoreMime]`

**HTTP Method**

POST

**HTTP Response**

201

**Request Content**

text/xml

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>{pid}</td>
<td>persistent identifier of the object to be created</td>
<td>new (see below)</td>
<td></td>
</tr>
<tr>
<td>new</td>
<td>indicator that either a new PID should be created for this object or that the PID to be used is encoded in the XML included as the body of the request</td>
<td>new</td>
<td></td>
</tr>
<tr>
<td>label</td>
<td>the label of the new object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>encoding</td>
<td>the encoding of the XML to be ingested. If this is specified, and given as anything other than UTF-8, you must ensure that the same encoding is declared in the XML. For example, if you specify &quot;ISO-8859-1&quot; as the encoding, the XML should start with: <code>&lt;xml version=&quot;1.0&quot; encoding=&quot;ISO-8859-1&quot;&gt;</code></td>
<td>UTF-8</td>
<td></td>
</tr>
<tr>
<td>namespace</td>
<td>the namespace to be used to create a PID for a new empty object; if object XML is included with the request, the namespace parameter is ignored</td>
<td>the default namespace of the repository</td>
<td></td>
</tr>
<tr>
<td>ownerId</td>
<td>the id of the user to be listed at the object owner</td>
<td></td>
<td></td>
</tr>
<tr>
<td>logMessage</td>
<td>a message describing the activity being performed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ignoreMime</td>
<td>indicates that the request should not be checked to ensure that the content is XML prior to attempting an ingest. This is provided to allow for client applications which do not indicate the correct Content-Type when submitting a request.</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>XML file as request content</td>
<td>file to be ingested as a new object</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Notes**

Executing this request with no request content will result in the creation of a new, empty object (with either the specified PID or a system-assigned PID). The new object will contain only a minimal DC datastream specifying the dc:identifier of the object.

**Examples**

POST: /objects/new

POST: /objects

POST: /objects/new?namespace=demo

POST: /objects/test:100?label=Test

**modifyDatastream**
URL Syntax
/objects/{pid}/datastreams/{dsID} ? [dsLocation] [altIDs] [dsLabel] [versionable] [dsState] [formatURI] [checksumType] [checksum] [mimeType] [logMessage] [ignoreContent] [lastModifiedDate]

HTTP Method
PUT

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pid)</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(dsID)</td>
<td>datastream identifier</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dsLocation</td>
<td>location of datastream content</td>
<td></td>
<td></td>
</tr>
<tr>
<td>altIDs</td>
<td>alternate identifiers for the datastream</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dsLabel</td>
<td>the label for the datastream</td>
<td></td>
<td></td>
</tr>
<tr>
<td>versionable</td>
<td>enable versioning of the datastream</td>
<td></td>
<td>true, false</td>
</tr>
<tr>
<td>dsState</td>
<td>one of &quot;A&quot;, &quot;I&quot;, &quot;D&quot; (&quot;A&quot;ctive, &quot;I&quot;nactive, &quot;D&quot;eleted)</td>
<td>A</td>
<td>A, I, D</td>
</tr>
<tr>
<td>formatURI</td>
<td>the format URI of the datastream</td>
<td></td>
<td></td>
</tr>
<tr>
<td>checksum</td>
<td>the algorithm used to compute the checksum</td>
<td>DEFAULT</td>
<td>DEFAULT, DISABLED, MD5, SHA-1, SHA-256, SHA-384, SHA-512</td>
</tr>
<tr>
<td>checksum</td>
<td>the value of the checksum represented as a hexadecimal string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mimeType</td>
<td>the MIME type of the content being added, this overrides the Content-Type request header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>logMessage</td>
<td>a message describing the activity being performed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ignoreContent</td>
<td>tells the request handler to ignore any content included as part of the request, indicating that you do not intend to update the datastream content. This is primarily provided to allow the use of client tools which always require content to be included as part of PUT requests.</td>
<td>false</td>
<td>true, false</td>
</tr>
<tr>
<td>lastModifiedDate</td>
<td>date/time of the last (known) modification to the datastream, if the actual last modified date is later, a 409 response is returned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>multipart file as request content</td>
<td>file to replace existing datastream (for Managed datastreams)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples
PUT: /objects/demo:35/datastreams/HIGH (with Multipart file)

modifyObject
URL Syntax
/objects/{pid} ? [label] [ownerId] [state] [logMessage] [lastModifiedDate]

HTTP Method
PUT

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>{pid}</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>label</td>
<td>the new object label</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ownerId</td>
<td>the id of the user to be listed at the object owner</td>
<td></td>
<td></td>
</tr>
<tr>
<td>state</td>
<td>the new object state - <em>A</em>ctive, <em>I</em>nactive, or <em>D</em>eleted</td>
<td>A</td>
<td>A, I, D</td>
</tr>
<tr>
<td>logMessage</td>
<td>a message describing the activity being performed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastModifiedDate</td>
<td>date/time of the last (known) modification to the datastream, if the actual last modified date is later, a 409 response is returned</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples
PUT: /objects/demo:29?label=Updated

purgeDatastream

URL Syntax
/objects/{pid}/datastreams/{dsID} ? [startDT] [endDT] [logMessage]

HTTP Method
DELETE

HTTP Response
200 with a string array of the date-time stamps of the versions purged

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>{pid}</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{dsID}</td>
<td>datastream identifier</td>
<td></td>
<td></td>
</tr>
<tr>
<td>startDT</td>
<td>the (inclusive) start date-time stamp of the range. If not specified, this is taken to be the lowest possible value, and thus, the entire version history up to the endDT will be purged</td>
<td>yyyy-MM-dd or yyyy-MM-ddTHH:mm:ssZ</td>
<td></td>
</tr>
<tr>
<td>endDT</td>
<td>the (inclusive) ending date-time stamp of the range. If not specified, this is taken to be the greatest possible value, and thus, the entire version history back to the startDT will be purged</td>
<td>yyyy-MM-dd or yyyy-MM-ddTHH:mm:ssZ</td>
<td></td>
</tr>
<tr>
<td>logMessage</td>
<td>a message describing the activity being performed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples
DELETE: /objects/demo:35/datastreams/HIGH

purgeObject
URL Syntax
(objects/{pid} ? [logMessage]

HTTP Method
DELETE

HTTP Response
204

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pid)</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>logMessage</td>
<td>a message describing the activity being performed</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples
DELETE: /objects/demo:29

purgeRelationship

URL Syntax
(objects/{pid}/relationships ? [subject] [predicate] [object] [isLiteral] [datatype]

HTTP Method
DELETE

HTTP Response
200

Return body
Text indicating if the relationship was successfully purged: true or false

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pid)</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>subject</td>
<td>subject of the relationship. Either a URI for the object or one of its datastreams</td>
<td>URI of this object</td>
<td></td>
</tr>
<tr>
<td>predicate</td>
<td>predicate of the relationship</td>
<td></td>
<td></td>
</tr>
<tr>
<td>object</td>
<td>object of the relationship</td>
<td></td>
<td></td>
</tr>
<tr>
<td>isLiteral</td>
<td>true if the object of the relationship is a literal, false if it is a URI</td>
<td>true, false</td>
<td></td>
</tr>
<tr>
<td>datatype</td>
<td>if the object is a literal, the datatype of the literal (optional)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples
DELETE /objects/demo:29/relationships?subject=info%3afedora%2fdemo%3a29%2fDC&predicate=http%3a%2f%2fwww.example.org%2frels%2fname&object=dublin%20core&isLiteral=true

setDatastreamState
URL Syntax
/objects/{pid}/datastreams/{dsID} ? [dsState]

HTTP Method
PUT

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>{pid}</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{dsID}</td>
<td>datastream identifier</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples
PUT: /objects/demo:35/datastreams/HIGH?dsState=D

setDataStreamVersionable

URL Syntax
/objects/{pid}/datastreams/{dsID} ? [versionable]

HTTP Method
PUT

HTTP Response
200

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>{pid}</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{dsID}</td>
<td>datastream identifier</td>
<td></td>
<td></td>
</tr>
<tr>
<td>versionable</td>
<td>enable versioning of the datastream</td>
<td>true</td>
<td>true, false</td>
</tr>
</tbody>
</table>

Examples
PUT: /objects/demo:35/datastreams/HIGH?versionable=false

Validate

URL Syntax
/objects/{pid}/validate ? [asOfDateTime]

HTTP Method
GET

HTTP Response
200 (OK) if the validation could be carried out (even if the object is not valid)
404 If some object or datastream could not be carried out
401 If the user credentials was insufficient
400 If the parameters are misformed
409 If one of the relevant objects were locked

500 If something else failed on the server

Return Body

XML, adhering to this schema
<xs:schema targetNamespace="http://www.fedora.info/definitions/1/0/access/
 xmlns="http://www.fedora.info/definitions/1/0/access/
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified">
 <xs:element name="validation">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="asOfDateTime"/>
    <xs:element ref="contentModels"/>
    <xs:element ref="problems"/>
    <xs:element ref="datastreamProblems"/>
   </xs:sequence>
   <xs:attribute name="pid" use="required">
    <xs:simpleType>
     <xs:restriction base="xs:string"/>
    </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="valid" use="required">
    <xs:simpleType>
     <xs:restriction base="xs:boolean"/>
    </xs:simpleType>
   </xs:attribute>
  </xs:complexType>
 </xs:element>
 <xs:element name="asOfDateTime">
  <xs:simpleType>
   <xs:restriction base="xs:dateTime"/>
  </xs:simpleType>
 </xs:element>
 <xs:element name="contentModels">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="model" minOccurs="0" maxOccurs="unbounded" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="problems">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="problem" minOccurs="0" maxOccurs="unbounded" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="datastreamProblems">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="datastream" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="datastream">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="problem" minOccurs="0" maxOccurs="unbounded" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="datastreamID" use="required">
  <xs:simpleType>
   <xs:restriction base="xs:string"/>
  </xs:simpleType>
 </xs:attribute>
 </xs:complexType>
 </xs:element>
</xs:schema>
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pid)</td>
<td>persistent identifier of the digital object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>asOfDateTime</td>
<td>indicates that the result should be relative to the digital object</td>
<td>yyyy-MM-dd or</td>
<td>yyyy-MM-ddTHH:mm:ssZ</td>
</tr>
<tr>
<td></td>
<td>and the repository as it existed at the given date and time</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Examples**

GET /objects/validate/demo:29?asOfDateTime=2008-01-01
Returns HTTP 200 with the body

```xml
<?xml version="1.0" encoding="UTF-8"?>
<validation pid="demo:29" valid="true">
  <asOfDateTime>2008-01-01T00:00:00.000Z</asOfDateTime>
  <contentModels>
    <model>info:fedora/fedora-system:FedoraObject-3.0</model>
  </contentModels>
  <problems>
  </problems>
  <datastreamProblems>
  </datastreamProblems>
</validation>
```

GET /objects/validate/demo:fail
Returns HTTP 200 with the body. Here the error was a mispelled element in the DC datastream, "dc:titel"

```xml
<?xml version="1.0" encoding="UTF-8"?>
<validation pid="demo:fail" valid="false">
  <asOfDateTime>fail</asOfDateTime>
  <contentModels>
    <model>info:fedora/fedora-system:FedoraObject-3.0</model>
  </contentModels>
  <problems>
  </problems>
  <datastreamProblems>
    <datastream datastreamID="DC">
  </datastream>
  </datastreamProblems>
</validation>
```

**Utility Methods**

**Upload**
URL Syntax
/upload

HTTP Method
POST

HTTP Response
202 and a URI for the uploaded file

Parameters
Multipart file as request content

Examples
POST: /upload (with Multipart file)

WADL
When running your own Fedora server, the REST API WADL is available at /objects/application.wadl.

Example:
http://localhost:8080/fedora/objects/application.wadl