RESTful HTTP API

Overview

Introduction

The Fedora 4 HTTP API is generally a RESTful API. HTTP methods like GET, PUT, POST and DELETE are implemented on most resource paths. The API also relies heavily on content negotiation to deliver context-appropriate responses, and a HATEOAS-driven text/html response (providing a decent GUI experience on top of the repository). The Fedora 4 RDF-based responses may be serialized as:

- application/ld+json
- application/n-triples
- application/rdf+xml
- text/n3 (or text/rdf+n3)
- text/plain
- text/turtle (or application/x-turtle)

The text/html response also includes embedded RDFa markup.

Fedora 4 implements the Linked Data Platform 1.0 Architecture, which:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)
4. Include links to other URIs, so that they can discover more things

Endpoints

Resources

Repository objects can be loosely divided into two classes of resources:

- Containers ("fedora:Container"), containing RDF properties and 0 or more child resources
- Binaries, containing any binary payload (roughly corresponding to Fedora 3 datastreams)

Containers

Request URI: /path/to/some/resource

Methods: GET, POST, PUT, PATCH, HEAD, OPTIONS, DELETE, MOVE, COPY

<table>
<thead>
<tr>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve the content of the resource</td>
</tr>
</tbody>
</table>

Request Headers:

- **RANGE**: Byte range of content to retrieve, of the form: "Range: bytes=500-999"
- **ACCEPT**: RDF sources support content negotiation with these formats: application/ld+json, application/n-triples, application/rdf+xml, application/x-turtle, application/xhtml+xml, application/xml, text/html, text/n3, text/plain, text/rdf+n3, text/turtle

Note: JSON-LD profiles can be requested by using the following Accept: headers:

- Accept: application/ld+json; profile="http://www.w3.org/ns/json-ld#expanded" - default
- Accept: application/ld+json; profile="http://www.w3.org/ns/json-ld#compacted"
- Accept: application/ld+json; profile="http://www.w3.org/ns/json-ld#flattened"

- **LIMIT**: Number of child resources to list. If paired with Accept: (X)HTML, the default is 100. Otherwise, the default is -1 (all children are listed)
- **IF-NONE-MATCH**
- **IF-MODIFIED-SINCE**
- **PREFER**
  - return=minimal
  - return=representation; include=" URIs "
  - return=representation; omit=" URIs "

where minimal indicates that only triples directly related to a resource should be returned, and representation indicates that links to other resources and their properties should also be included. URIs is a space-separate list of LDP defined preferences, and also:

<table>
<thead>
<tr>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://fedora.info/definitions/v4/repository#EmbedResources">http://fedora.info/definitions/v4/repository#EmbedResources</a></td>
<td>Embed &quot;child&quot; resources in the returned representation</td>
</tr>
<tr>
<td><a href="http://fedora.info/definitions/v4/repository#InboundReferences">http://fedora.info/definitions/v4/repository#InboundReferences</a></td>
<td>Include assertions from other Fedora resources to this node (excluded from representation by default)</td>
</tr>
<tr>
<td><a href="http://fedora.info/definitions/v4/repository#ServerManaged">http://fedora.info/definitions/v4/repository#ServerManaged</a></td>
<td>Embed server managed properties in the representation (enabled by default)</td>
</tr>
</tbody>
</table>

Example (1): Turtle

```
curl -H "Accept: text/turtle" "http://localhost:8080/rest/path/to/resource"
```
Response:

Status: 200 OK

Headers:
ETag: "0ed38fe21c2663ace5322e970e7b7d606196e00"
Last-Modified: Mon, 19 May 2014 19:44:59 GMT
Preference-Applied: return=representation; include="http://www.w3.org/ns/ldp#PreferMembership http://www.w3.org/ns/ldp#PreferContainment"
Vary: Prefer
Accept-Patch: application/sparql-update
Allow: MOVE,COPY,DELETE,POST,HEAD,GET,PUT,PATCH,OPTIONS
Accept-Post: text/turtle,text/rdf+n3,application/n3,text/n3,application/rdf+xml,application/n-triples,multipart/form-data,application/sparql-update
Link: <http://www.w3.org/ns/ldp#Resource>;rel="type"
Link: <http://www.w3.org/ns/ldp#DirectContainer>;rel="type"
Content-Type: text/turtle

Body:

```
<http://localhost:8080/rest/path/to/resource> <http://fedora.info/definitions/v4/rest-api#hasAccessRoles>
<http://localhost:8080/rest/path/to/resource/fcr:accessroles> ;
  <http://fedora.info/definitions/v4/repository#hasParent> <http://localhost:8080/rest/path/to> ;
  a <http://www.w3.org/ns/ldp#Container> , <http://www.w3.org/ns/ldp#DirectContainer> ;
  <http://www.w3.org/ns/ldp#membershipResource> <http://localhost:8080/rest/path/to/resource> ;
  <http://www.w3.org/ns/ldp#hasMemberRelation> <http://fedora.info/definitions/v4/repository#hasChild> ;
  <http://fedora.info/definitions/v4/repository#primaryType> "nt:folder"^^<http://www.w3.org/2001/XMLSchema#string> ;
  <http://fedora.info/definitions/v4/repository#lastModifiedBy> "bypassAdmin"^^<http://www.w3.org/2001/XMLSchema#string> ;
  <http://fedora.info/definitions/v4/repository#uuid> "2b655d37-e88a-44c5-86c1-3b3935cdea49"^^<http://www.w3.org/2001/XMLSchema#string> ;
  <http://fedora.info/definitions/v4/repository#createdBy> "bypassAdmin"^^<http://www.w3.org/2001/XMLSchema#string> ;
```

Example (2): RDF/XML

```
curl -H "Accept: application/rdf+xml" "http://localhost:8080/rest/path/to/resource"
```

Response:

Status: 200 OK

Headers:
ETag: "0ed38fe21c2663ace5322e970e7b7d606196e00"
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about="http://localhost:8080/rest/path/to/resource">
        <rdf:type rdf:resource="http://www.w3.org/ns/ldp#Container"/>
        <rdf:type rdf:resource="http://www.w3.org/ns/ldp#DirectContainer"/>
        <hasMemberRelation xmlns="http://www.w3.org/ns/ldp#" rdf:resource="http://fedora.info/definitions/v4/repository#hasChild"/>
        <rdf:type rdf:resource="http://www.jcp.org/jcr/nt/1.0folder"/>
        <rdf:type rdf:resource="http://www.jcp.org/jcr/nt/1.0hierarchyNode"/>
        <rdf:type rdf:resource="http://www.jcp.org/jcr/mix/1.0base"/>
        <mixinTypes xmlns="http://fedora.info/definitions/v4/repository#" rdf:datatype="http://www.w3.org/2001/XMLSchema#string">fedora:resource</mixinTypes>
        <mixinTypes xmlns="http://fedora.info/definitions/v4/repository#" rdf:datatype="http://www.w3.org/2001/XMLSchema#string">fedora:object</mixinTypes>
        <uuid xmlns="http://fedora.info/definitions/v4/repository#" rdf:datatype="http://www.w3.org/2001/XMLSchema#string">2b655d37-e88a-44c5-86c1-3b3935cdea49</uuid>
        <primaryType xmlns="http://fedora.info/definitions/v4/repository#" rdf:datatype="http://www.w3.org/2001/XMLSchema#string">nt:folder</primaryType>
        <xml:namespace prefix="jcr:nt" namespace="http://www.jcp.org/jcr/nt/1.0"/>
        <xml:namespace prefix="jcr:mix" namespace="http://www.jcp.org/jcr/mix/1.0"/>
        <xml:namespace prefix="jcr" namespace="http://www.jcp.org/jcr/1.0"/>
        <mixinTypes xmlns="http://fedora.info/definitions/v4/repository#" rdf:datatype="http://www.w3.org/2001/XMLSchema#string">fedora:resource</mixinTypes>
        <mixinTypes xmlns="http://fedora.info/definitions/v4/repository#" rdf:datatype="http://www.w3.org/2001/XMLSchema#string">fedora:object</mixinTypes>
        <uuid xmlns="http://fedora.info/definitions/v4/repository#" rdf:datatype="http://www.w3.org/2001/XMLSchema#string">2b655d37-e88a-44c5-86c1-3b3935cdea49</uuid>
        <primaryType xmlns="http://fedora.info/definitions/v4/repository#" rdf:datatype="http://www.w3.org/2001/XMLSchema#string">nt:folder</primaryType>
    </rdf:Description>
</rdf:RDF>
Example (3): Prefer headers

```
curl -H "Prefer: return=representation; include="http://fedora.info/definitions/v4/repository#InboundReferences"; omit="http://www.w3.org/ns/ldp#PreferMembership http://www.w3.org/ns/ldp#PreferContainment"" http://localhost:8080/rest/parent
```

Response:
Example (4): Non-RDF Source

curl "http://localhost:8080/rest/path/to/binary/resource"

Response:
POST Create new resources within a LDP container

Request Headers:

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTENT-DISPOSITION</td>
<td>(Optional) The filename provided in the content disposition header will be stored in a ebucore:filename property. (See 'Example (6): Uploaded file with filename')</td>
</tr>
<tr>
<td>CONTENT-TYPE</td>
<td>(Optional) MIME type of the uploaded binary or RDF content, including: message/external-body (See 'Example (2): Create a new child binary resource with empty content')</td>
</tr>
</tbody>
</table>

If the MIME type corresponds to a supported RDF format or SPARQL-Update, the uploaded content will be parsed as RDF and used to populate the child node properties. RDF will be interpreted using the current resource as the base URI (e.g. <> will be expanded to the current URI). Namespaces must be declared in full.

For other MIME types, the uploaded content will be used to create a binary resource.

<table>
<thead>
<tr>
<th>Slug</th>
<th>(Optional) A suggested name for the new child resource, which the repository may ignore.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIGEST</td>
<td>(Optional) A way of providing one or more checksums (SHA-1, SHA-256, MD5) which will be checked against the uploaded content to ensure error-free transfer. The header is interpreted according to RFC-3230 (See 'Example (4): Uploaded file with checksum' and 'Example (5): Uploaded file with checksum mismatch') If the provided checksum does not match the calculated value for the uploaded file a 409 Conflict will be returned. See PATC H example for documentation on how to change the default algorithm used during on-demand fixity checking.</td>
</tr>
</tbody>
</table>

Example (1): Create a new child node

curl -X POST "http://localhost:8080/rest/"
Example (2): Create a new child binary resource with empty content

curl -i -X POST -H "Content-Type:text/plain" "http://localhost:8080/rest"

Response:

Status: 201 Created
ETag: "487f188240d2be32ae32d49958c0e1ea18224be0a"
Last-Modified: Thu, 29 May 2014 15:46:13 GMT
Link: <http://localhost:8080/rest/b8/fc/32/be/b8fc32be-34be-428e-bf7c-81be97e5f2e3/fcr:metadata>; rel="describedby";
anchor="http://localhost:8080/rest/b8/fc/32/be/b8fc32be-34be-428e-bf7c-81be97e5f2e3"
Location: http://localhost:8080/rest/b8/fc/32/be/b8fc32be-34be-428e-bf7c-81be97e5f2e3
Content-Type: text/plain
Transfer-Encoding: chunked

Body:
http://localhost:8080/rest/b8/fc/32/be/b8fc32be-34be-428e-bf7c-81be97e5f2e3

Example (3): Create a new container with RDF properties


Request Body:
PREFIX dc: <http://purl.org/dc/elements/1.1/>
<> dc:title "some-resource-title"

Response:

Status: 201 Created
ETag: "a66937ca13bc29c63f7b1a55d447a76cd8f1d573"
Location: http://localhost:8080/rest/new/node/uri

Body:
http://localhost:8080/rest/new/node/uri

Example (4): Uploaded file with checksum

curl -i -X POST --data-binary "@picture.jpg" -H "digest: sha1=cb1a576f22e8e3e110611b616a3e2f5ce9bd941" "http://localhost:8080/rest/parent/container"

Response:
Example (4b): Uploaded file with multiple checksum

```bash
curl -i -X POST --data-binary @picture.jpg -H"digest: sha1=cb1a576f22e3e3e110611b616e3e2f5ce9b5b941,
sha256=95331b6710d0400ee4913b5613c78f54" "http://localhost:8080/rest/parent/container"
```

Response:

```
Status: 201 Created

Headers:
ETag: "fe4e2bf389fe05dab6390564e40012d8d36eca19"
Location: http://localhost:8080/rest/parent/object/auto/generated/ds/id

Body:
http://localhost:8080/rest/parent/object/auto/generated/ds/id
```

Example (5): Uploaded file with checksum mismatch

```bash
curl -X POST --data-binary @picture.jpg -H"digest: sha1=checksumdoesntmatch" "http://localhost:8080/rest/parent/object"
```

Response:

```
Status: 409 Conflict

Body:
Checksum Mismatch of cb1a576f22e3e3e110611b616e3e2f5ce9b5b941 and urn:sha1:checksumdoesntmatch
```

Example (6): Uploaded file with filename

```bash
curl -i -X POST --data-binary @picture.jpg -H "Content-Disposition: attachment; filename="picture.jpg"\n" "http://localhost:8080/rest/parent/container"
```

Response:

```
Status: 201 Created

Headers:
ETag: "fe4e2bf389fe05dab6390564e40012d8d36eca19"
Location: http://localhost:8080/rest/parent/object/auto/generated/ds/id

Body:
http://localhost:8080/rest/parent/object/auto/generated/ds/id
```
### Status:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Created</td>
</tr>
<tr>
<td>404</td>
<td>Not Found (resource does not exist – use PUT to specify a new resource name)</td>
</tr>
<tr>
<td>409</td>
<td>Conflict (checksum mismatch)</td>
</tr>
</tbody>
</table>

### PUT

Create a resource with a specified path, or replace the triples associated with a resource with the triples provided in the request body.

Some resource properties are repository managed and cannot be removed or modified. Any attempt to do so may result in a 4xx error. RDF will be interpreted using the current resource as the base URI (e.g. <> will be expanded to the current URI).

### Request Headers:

<table>
<thead>
<tr>
<th>Header</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTENT-TYPE</td>
<td>text/turtle, text/rdf+n3, application/n3, text/n3, application/rdf+xml, application/n-triples, text/html, text/plain, application/ld+json, message/external-body</td>
</tr>
<tr>
<td>IF-MATCH</td>
<td></td>
</tr>
<tr>
<td>DIGEST</td>
<td>(Optional) A way of providing one or more checksums (SHA-1, SHA-256, MD5) which will be checked against the uploaded content to ensure error-free transfer. The header is interpreted according to RFC-3230 (See 'Example (3): Creating new binary resource at a specified path') If the provided checksum does not match the calculated value for the uploaded file a 409 Conflict will be returned. See Fixity Service for documentation on how to change the default algorithm used during on-demand fixity checking.</td>
</tr>
<tr>
<td>PREFER</td>
<td>With the value handling=lenient; received=&quot;minimal&quot; allows replacing the properties of a container without having to provide all of the server-managed triples.</td>
</tr>
</tbody>
</table>

### Example (1): Updating properties with RDF content

1.1 Perform a GET to the current container's RDF content

```
```

Body:
```
PREFIX dc: <http://purl.org/dc/elements/1.1/>

PREFIX ldp: <http://www.w3.org/ns/ldp#>
fedora:primaryType "nt:folder"^^<http://www.w3.org/2001/XMLSchema#string> ;
<http://fedora.info/definitions/v4/repository#jcr/xml> rdfs:label "jcr/xml"^^<http://www.w3.org/2001/XMLSchema#string> ;
```

1.2 Add your updates (e.g. <> dc:title "some-resource-title" .) under the full result of the GET response in 1.1

Request Body:

PREFIX dc: <http://purl.org/dc/elements/1.1/>

...  
PREFIX ldp: <http://www.w3.org/ns/ldp#>
   fcrepo:primaryType "nt:folder"^^<http://www.w3.org/2001/XMLSchema#string> ;  
   <http://fedora.info/definitions/v4/repository#jcr/xml> rdfs:label "jcr/xml"^^<http://www.w3.org/2001/XMLSchema#string> ;  
<> dc:title "some-resource-title" .

Response:

Status: 204 No Content

Note that to prevent conflicts, the value of the `fedora:lastModified` triple in the request must match its current counterpart in the repository. The way to ensure this is to base the changes in a PUT request on the RDF content obtained in an immediately prior GET request, and never try to PUT consecutively, as this will result in a 400 error.

Example (2): Creating a new resource at a specified path

```bash
curl -i -X PUT "http://localhost:8080/rest/node/to/create"
```

Response:

Status: 201 Created

Response Headers:
ETag: "ef214795c3b9109389f5a542a6c081976e9a9587"  
Last-Modified: Mon, 19 May 2014 20:52:57 GMT  
Location: http://localhost:8080/rest/node/to/create

Body:
http://localhost:8080/rest/node/to/create

Example (3): Creating new binary resource at a specified path

```bash
```

Response:
Status: 201 Created

Headers:
ETag: "urn:sha1:ea3d29b28522724a82de042a7b7a4e79a1653435"
Last-Modified: Mon, 19 May 2014 20:55:31 GMT
Location: http://localhost:8080/rest/new/image

Body:
http://localhost:8080/rest/new/image

Note that once a resource is created as an RDF or non-RDF source, it must remain an RDF or non-RDF source. In order to change the type of resource, you must DELETE and then re-create the resource.

Status:

204 No Content
412 Precondition Failed

Example (4): Creating a new binary resource at a specified path redirecting to external content

curl -X PUT -H "Content-Type: message/external-body; access-type=URL; URL="http://www.example.com/file"" "http://localhost:8080/rest/node/to/create"

Response:

Status: 201 Created

Headers:
ETag: "ef214795c3b9109389ffa542a6c881976e1e9587"
Last-Modified: Mon, 19 May 2014 20:52:57 GMT
Link: <http://localhost:8080/rest/node/to/create/fcr:metadata>; rel="describedby"
Location: http://localhost:8080/rest/node/to/create

Body:
http://localhost:8080/rest/node/to/create

Modify the triples associated with a resource with SPARQL-Update

Request Headers:

IF-MATCH
IF-UNMODIFIED-SINCE
CONTENT-TYPE

Example (1):

Request Body:

PREFIX dc: <http://purl.org/dc/elements/1.1/>
INSERT {
  <> dc:title "some-resource-title" .
}
WHERE {}

Response:

Status: 204 No Content

Example (2) Update default fixity algorithm:


Request Body:

PREFIX fedoraconfig: <http://fedora.info/definitions/v4/config#>
INSERT {
  <> fedoraconfig:defaultDigestAlgorithm "sha-256"
}
WHERE {}

Response:

Status: 204 No Content

Status:

| 204 | No Content |
| 412 | Precondition Failed |

DELETE Delete a resource

Example:

curl -X DELETE "http://localhost:8080/rest/resource/to/delete"

Response:

Status: 204 No Content

Delete requests create "tombstone" resources which need to be removed before you can reuse a URI.

Example:
curl -X GET "http://localhost:8080/rest/some/deleted/resource"

Status: 410 Gone
Link: <http://localhost:8080/rest/some/deleted/resource/fcr:tombstone>; rel="hasTombstone"

Body:
Discovered tombstone resource at /some/deleted/resource, departed: 2016-11-14T17:26:17.334-06:00

curl -X DELETE "http://localhost:8080/rest/some/deleted/resource/fcr:tombstone"

Status: 204 No Content

Create a new node and reuse the deleted URL

curl -X PUT "http://localhost:8080/rest/some/deleted/resource"

Status: 201 Created

Status:
- 204 No Content
- 404 Not Found

HEAD Retrieve the resource headers

Example:
curl -i -X HEAD "http://localhost:8080/rest/resource/to/head"

Response:
Status: 200 OK
ETag: "bbdd92e395800153a686773f773bcad80a51f47b"
Last-Modified: Wed, 28 May 2014 18:31:36 GMT
Link: <http://www.w3.org/ns/ldp#Resource>;rel="type"
Link: <http://www.w3.org/ns/ldp#Container>;rel="type"
Link: <http://www.w3.org/ns/ldp#BasicContainer>;rel="type"
Accept-Patch: application/sparql-update
Accept-Post: text/turtle,text/rdf+n3,text/n3,application/rdf+xml,application/n-triples,multipart/form-data,
application/sparql-update
Allow: MOVE,COPY,DELETE,POST,HEAD,GET,PUT,PATCH,OPTIONS

Status:
- 200 OK
- 404 Not Found
**OPTIONS**

Outputs information about the supported HTTP methods, etc.

**Example:**

```
curl -i -X OPTIONS "http://localhost:8080/rest/node/to/options"
```

**Response:**

```
Status: 200 OK
Accept-Patch: application/sparql-update
Allow: MOVE,COPY,DELETE,POST,HEAD,GET,PUT,PATCH,OPTIONS
Accept-Post: text/turtle,text/rdf+n3,application/n3,text/n3,application/rdf+xml,application/n-triples,multipart/form-data,application/sparql-update
```

**Status:**

```
200 OK
```

---

**MOVE**

**Move a resource (and its subtree) to a new location**

Note: A federated resource can be moved within the federation, but not from the federation to the internal repository (or vice versa)

**Example (1):**

```
```

**Response:**

```
Status: 201 Created
Headers:
Location: http://localhost:8080/rest/new/path/to/resource
```

**Example (2):**

```
curl -X MOVE "http://localhost:8080/rest/path/to/resource"
```

**Response:**

```
Status: 502 Bad Gateway
```

**Status:**

```
201 Created
409 Source path doesn't exist
412 Destination path already exists
502 Destination URI isn't a valid resource path
```
Copy a resource (and its subtree) to a new location

Example:


Response:

Status: 201 Created

Headers:
Location: http://localhost:8080/rest/new/path/to/resource

Status:

- **201**: Created
- **409**: Source path doesn't exists
- **412**: Destination path already exists
- **502**: Destination URI isn't a valid resource path

Versioning

Request URI: `/path/to/some/resource/fcr:versions`

Methods: GET, POST, PATCH

Get a list of the available versions of an object

Request Headers:

- **ACCEPT**: application/id+json, application/n-triples, application/rdf+xml, application/x-turtle, application/xhtml+xml, application/xml, text/html, text/n3, text/plain, text/rdf+n3, text/turtle

Example:


Response:
Status: 200 OK

Headers:
Content-Type: text/turtle

Body:
<http://localhost:8080/rest/path/to/resource> fedora:hasVersion <http://localhost:8080/rest/path/to/resource/fcr:versions/87a0a8c317f1e749515d33-cb73-4fd7-9d1d-c715eb6947e0> .
<http://localhost:8080/rest/path/to/resource/fcr:versions/87a0a8c317f1e749515d33-cb73-4fd7-9d1d-c715eb6947e0> fedora:hasVersionLabel "v0"^^<http://www.w3.org/2001/XMLSchema#string> ;

<http://localhost:8080/rest/path/to/resource/fcr:versions/87a0a8c317f1e7dae533ec-b1b6-4dal-8bb9-f1964e253572> fedora:hasVersionLabel "v1"^^<http://www.w3.org/2001/XMLSchema#string> ;

Status:

200 OK

404 Not Found: This resource is not versioned or this resource is not exist.

Response: The requested resource is not available.

---

**GET**

Get a previous version of an object

Request Headers:

| ACCEPT | application/id+json, application/n-triples, application/rd+xml, application/x-turtle, application/xhtml+xml, application/xml, text/html, text/n3, text/plain, text/rd+nv3, text/turtle |

Example:

```bash
```

Response:
POST  Create a new version of an object

Request Headers

<table>
<thead>
<tr>
<th>SLUG</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Required) A suggested name for the new child resource, which the repository may ignore.</td>
</tr>
</tbody>
</table>

Example:

```bash
```

Response:

| Status: 201 Created |

Status:

| 200 | OK |
|     |    | 404 | Not Found: if the version or the resource does not exist |

Response: The requested resource is not available.
Conflict: if the version label is already in use for another version of this resource

Response: The requested resource is not available.

---

**PATCH**

Revert to a previous version of an object

Example:

```
curl -X PATCH http://localhost:8080/rest/path/to/resource/fcr:versions/existingVersionName
```

Response:

```
Status: 204 No Content
```

Status:

- **204 No Content**: if the version is reverted successfully
- **404 Not Found**: if the version does not exist

Response: The requested resource is not available.

---

Request URI: `/path/to/some/resource/fcr:versions/versionName`

Methods: DELETE

**DELETE**

Remove a previous version of an object

Example:

```
curl -X DELETE http://localhost:8080/rest/path/to/resource/fcr:versions/versionName
```

Response:

```
Status: 204 No Content
```

Trying to delete the current version of a resource will result in an error. To revert to an old version of a resource, first revert to the old version using the PATCH method, and then delete the newer version.

Status:

- **204 No Content**: if the version is reverted successfully
- **400 Bad Request**: if trying to delete the most recent version

Response: Cannot remove current version

- **404 Not Found**: if the version does not exist

Response: The requested resource is not available.
Backup and Restore

Request URI: / 
Methods: GET, POST

**POST fcr:backup Initiate a (consistent) backup of the repository**
You may include a backup directory in the body of the POST request. If you do not specify a directory, the backup will be stored in a temporary directory on the server (which will be returned in the response body). As yet, there is not a REST API method for retrieving the content of the backup.

Example:
```
curl -X POST "http://localhost:8080/rest/fcr:backup"
```

Response:
```
Status: 200 OK
Body: /tmp/fcrepo4-data/path/to/backup/directory
```

Example 2: Including a destination directory
```
curl -X POST -d /path/to/backup/destination "http://localhost:8080/rest/fcr:backup"
```

Response:
```
Status: 200 OK
```

**POST fcr:restore Replace repository content with data from a backup directory**

Example:
```
```

Response:
Status: 204 No Content

Status:

- **204**: Backup restored
- **500**: Error restoring backup

Fixity

Request URI: `/path/to/some/resource/fcr:fixity`

Methods: GET

Get the fixity report for an object

Checking fixity requires retrieving the content from the binary store and may take some time.

Request Headers:

- `ACCEPT application/ld+json, application/n-triples, application/rdf+xml, application/x-turtle, application/xhtml+xml, application/xml, text/html, text/n3, text/plain, text/rdf+n3, text/turtle`

Example:

```
```

Response (fixity success):

Status: 200 OK

Readers:

Content-Type: text/turtle

Body:

```turtle
@prefix premis: <http://www.loc.gov/premis/rdf/v1#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<http://localhost:8080/rest/path/to/some/resource>
  premis:hasFixity <http://localhost:8080/rest/path/to/some/resource#fixity/1494431303920> .

<http://localhost:8080/rest/path/to/some/resource#fixity/1494431303920>
  rdf:type premis:Fixity ;
  rdf:type premis:EventOutcomeDetail ;
  premis:hasEventOutcome "SUCCESS" ;
  premis:hasMessageDigestAlgorithm "SHA-1" ;
  premis:hasMessageDigest <urn:sha1:ca3392593351ef8e6554bdabfbdbdc1002ecb6f> ;
  premis:hasSize "1277811"^^<http://www.w3.org/2001/XMLSchema#long> .

```

Response (fixity failure):
Default Fixity Algorithm

When ingesting a binary resource, one or more checksums may be provided, see API reference (POST example 4b). The supported algorithms are: SHA-1, SHA-256 and MD5.

By default, the algorithm used by the Fixity service is SHA-1. However, that may be changed to one of the other supported algorithms by configuring the property: `fedoraconfig:defaultDigestAlgorithm`. See API reference (PATCH example 2).

Transactions


Methods: GET, POST

GET Get the current status of the repository in a transaction

Example:

curl -i "http://localhost:8080/rest/tx:86dd0891-d975-42d8-8837-a24ad6041b59"

Response:

Status: 200 OK

Link: <http://localhost:8080/rest/>;rel="canonical"

Body:
Same as non-transaction response.
### POST Create a new transaction

After retrieving a transaction resource, the client can execute any REST API method prefixed by the transaction location. Transactions are automatically closed and rolled back after 3 minutes of inactivity. Transactions can be refreshed by POSTing to `/rest/{tx:id}/fcr:tx`

Example:

```
curl -i -X POST "http://localhost:8080/rest/fcr:tx"
```

Response:

<table>
<thead>
<tr>
<th>Status</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>201 Created</td>
<td>Created: if the transaction is created successfully</td>
</tr>
</tbody>
</table>

Usage:

When a transaction has been created, it will return a Location header. Use this location as the base URL for performing REST API operations within the transaction. When you are done with the transaction, either append `"fcr:tx/fcr:commit"` to commit the transaction to the repository or `"fcr:tx/fcr:rollback"` to discard the changes.

```
curl -X POST "http://localhost:8080/rest/tx:83e34464-144e-43d9-af13-b3464a1fb9b5/path/to/object/to/create"
curl -X DELETE "http://localhost:8080/rest/tx:83e34464-144e-43d9-af13-b3464a1fb9b5/path/to/resource/to/delete"
```

### POST Keep an existing transaction alive

Example:

```
```

Response:

<table>
<thead>
<tr>
<th>Status</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>204 No Content</td>
<td>No Content: if the transaction is renewed successfully</td>
</tr>
</tbody>
</table>

Usage:

Use this Location header as the base URL for performing REST API operations within the transaction. When you are done with the transaction, add `/fcr:tx/fcr:commit` to commit the transaction to the repository or `/fcr:tx/fcr:rollback` to discard the changes.

```
curl -X POST "http://localhost:8080/rest/tx:83e34464-144e-43d9-af13-b3464a1fb9b5/path/to/object/to/create"
curl -X DELETE "http://localhost:8080/rest/tx:83e34464-144e-43d9-af13-b3464a1fb9b5/path/to/resource/to/delete"
```
Save and commit an open transaction

Any operations you made within the scope of the transaction will be applied together, meaning if any of them fail, the whole transaction will fail.

Example:


Response:

Status: 204 No Content

Status:

| 204 | No Content: if the transaction is committed successfully |
| 410 | Gone: if the transaction doesn't exist |

Rollback and close an open transaction

Example:


Response:

Status: 204 No Content

Status:

| 204 | No Content: if the transaction is discarded successfully |
| 410 | Gone: if the transaction doesn't exist |