RESTful HTTP API

Table of Contents

- Overview
  - Introduction
- Endpoints
  - Resources
    - Containers
      - GET Retrieve the content of the resource
      - POST Create new resources within a LDP container
      - PUT Create a resource with a specified path, or replace the binary content or triples associated with a resource with the binary content or triples provided in the request body.
      - PATCH Modify the triples associated with a resource with SPARQL-Update
      - DELETE Delete a resource
      - HEAD Retrieve the resource headers
      - OPTIONS Outputs information about the supported HTTP methods, etc.
      - MOVE Move a resource (and its subtree) to a new location
      - COPY Copy a resource (and its subtree) to a new location
    - Versioning
      - Services
        - Backup and Restore
          - POST fcr:backup Initiate a (consistent) backup of the repository
          - POST fcr:restore Replace repository content with data from a backup directory
        - Fixity
          - HEAD Request the fixity checksum for a given digest algorithm
          - GET Get the fixity report for an object
          - Note: Default Fixity Algorithm
        - Transactions
          - GET Get the current status of the repository in a transaction
          - POST Create a new transaction
          - POST Keep an existing transaction alive
          - POST Save and commit an open transaction
          - POST Rollback and close an open transaction

Overview

Introduction

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

[...] describes the use of HTTP for accessing, updating, creating and deleting resources from servers that expose their resources as Linked Data. It provides clarifications and extensions of the rules of Linked Data [LINKED-DATA]:

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
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

### RESTful HTTP API - Containers

Request URI: `/path/to/some/resource`
Methods: GET, POST, PUT, PATCH, HEAD, OPTIONS, DELETE, MOVE, COPY

#### GET

Retrieve the content of the resource

**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` or `return=representation; include="URIs"` or `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/fcrepo#PreferInboundReferences">http://fedora.info/definitions/fcrepo#PreferInboundReferences</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>
<tr>
<td><a href="http://www.w3.org/ns/oa#PreferContainedDescriptions">http://www.w3.org/ns/oa#PreferContainedDescriptions</a></td>
<td>Embed &quot;child&quot; resources in the returned representation</td>
</tr>
<tr>
<td><a href="http://www.w3.org/ns/ldp#PreferContainment">http://www.w3.org/ns/ldp#PreferContainment</a></td>
<td>Include/Exclude &quot;ldp:contains&quot; assertions to contained resources (enabled by default)</td>
</tr>
<tr>
<td><a href="http://www.w3.org/ns/ldp#PreferMembership">http://www.w3.org/ns/ldp#PreferMembership</a></td>
<td>Include/Exclude assertions to member resources established by the Direct and Indirect containers (enabled by default)</td>
</tr>
<tr>
<td><a href="http://www.w3.org/ns/ldp#PreferMinimalContainer">http://www.w3.org/ns/ldp#PreferMinimalContainer</a></td>
<td>Include/Exclude triples that would be present when the container is empty (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: "0ed38fe211c2663ace5322e970e7b7d606196e00"
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,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:

例1：

```turtle
```

Example (2): RDF/XML

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

Response:

Status: 200 OK

Headers:
ETag: "0ed38fe211c2663ace5322e970e7b7d606196e00"
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,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:

例2：

```xml
```

Example (2): RDF/XML
Response:

Status: 200 OK

<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"/>
        <hasMemberRelation xmlns="http://www.w3.org/ns/ldp#" rdf:resource="http://fedora.info/definitions/v4/repository#hasChild"/>
        <rdf:resource rdf:resource="http://www.jcp.org/jcr/nt/1.0folder"/>
        <rdf:resource rdf:resource="http://www.jcp.org/jcr/nt/1.0base"/>
        <rdf:resource rdf:resource="http://www.jcp.org/jcr/mix/1.0lastModified"/>
        <mixinTypes xmlns="http://fedora.info/definitions/v4/repository#" rdf:resource="fedora:resource"/>
        <mixinTypes xmlns="http://fedora.info/definitions/v4/repository#" rdf:resource="fedora:object"/>
        <mixinTypes xmlns="http://fedora.info/definitions/v4/repository#" rdf:resource="fedora:relations"/>
        <lastModifiedBy xmlns="http://fedora.info/definitions/v4/repository#" rdf:resource="http://fedora.info/definitions/v4/repository#"/>
        <primaryType xmlns="http://fedora.info/definitions/v4/repository#" rdf:resource="http://www.w3.org/2001/XMLSchema#nt:folder"/>
        <lastModified xmlns="http://fedora.info/definitions/v4/repository#" rdf:resource="http://www.w3.org/2001/XMLSchema#"/>
        <createdBy xmlns="http://fedora.info/definitions/v4/repository#" rdf:resource="http://fedora.info/definitions/v4/repository#"/>
    </rdf:Description>
</rdf:RDF>
Example (3): Prefer headers

include="http://fedora.info/definitions/fcrepo#PreferInboundReferences"; omit="http://www.w3.org/ns/ldp#PreferMembership
http://www.w3.org/ns/ldp#PreferContainment"'

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:

- **CONTENT-DISPOSITION** *(Optional)* The filename provided in the content disposition header will be stored in an ebruco:filename property. (See 'Example (6): Uploaded file with filename')

- **CONTENT-TYPE** *(Optional)* MIME type of the uploaded binary or RDF content. (See 'Example (2): Create a new child binary resource with empty content')

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.

If the MIME type corresponds to a supported RDF format or SPARQL-Update, and you also provide a Link: <http://www.w3.org/ns/ldp#NonRDFSource>; rel="type" header. Then the contents will be stored as a binary resource with the provided MIME type as Content-type.

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

**Supported RDF formats**: text/turtle, text/rdf+n3, application/n3, text/n3, application/rdf+xml, application/n-triples, text/html, text/plain, application/json

- **SLUG** *(Optional)* A suggested name for the new child resource, which the repository may ignore.

The slug you provide does not guarantee the location of the created resource. Clients must check the Location header for the path to the created resource.

- **DIGEST** *(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 PATCH example for documentation on how to change the default algorithm used during on-demand fixity checking. Valid 'Digest' algorithms/values are: sha, sha-256, or md5.

- **LINK** With a value containing subfield rel= http://fedora.info/definitions/fcrepo#ExternalContent the client may provide binary content from an external path. See the external content documentation for details about syntax and handling options (note, this is disabled by default), and Example 7 for usage.

**Example (1): Create a new child node**
<table>
<thead>
<tr>
<th>Curl Command</th>
<th>Description</th>
</tr>
</thead>
</table>
| `curl -X POST "http://localhost:8080/rest/"` | Response:  
  Status: 201 Created  
  Headers:  
  ETag: "3ac31b09de1536a312d9c6ee845369947721840"  
  Location: http://localhost:8080/rest/some/path/to/a/new/resource  
  Body: http://localhost:8080/rest/some/path/to/a/new/resource |
| `curl -i -X POST -H "Content-Type:text/plain" "http://localhost:8080/rest"` | Response:  
  Status: 201 Created  
  ETag: "487f188240d2be3ae32d49958c0e8e18224be0a"  
  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 |
| `curl -i -X POST -H "Content-Type:text/turtle" --data-binary @body.rdf "http://localhost:8080/rest/"` | Request Body:  
  PREFIX dc: <http://purl.org/dc/elements/1.1/>  
  <> dc:title "some-resource-title"  
  Response:  
  Status: 201 Created  
  Headers:  
  ETag: "a66937ca13bc29c63f7b1a55d447a76cd8f1d573"  
  Location: http://localhost:8080/rest/new/node/uri  
  Body: http://localhost:8080/rest/new/node/uri |
| `curl -i -X POST --data-binary @picture.jpg -H "digest: sha=cbla576f22e8e3e110611b616a3e2f5ce9bdb941" "http://localhost:8080/rest/parent/container"` | Example (4): Uploaded file with checksum |
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 (4b): Uploaded file with multiple checksum

curl -i -X POST --data-binary "@picture.jpg" -H"digest: sha=cb1a576f22e8e3e110611b616e3e2f5ce9bdb941, sha-256=95311b60710d0400ee4913b5613c78f54" "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


Response:

Status: 409 Conflict

Body:
Checksum Mismatch of cb1a576f22e8e3e110611b616e3e2f5ce9bdb941 and urn:sha1:checksumdoesntmatch

Example (6): Uploaded file with filename


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 (7): Creating a new binary resource using proxied external content

```bash
curl -i -X POST -H"Link: <file:///path/to/picture.jpg>; rel="http://fedora.info/definitions/fcrepo#ExternalContent"; handling="proxy"; type="image/jpg"" "http://localhost:8080/rest/parent/container"
```

Response:

Status: 201 Created

Headers:
ETag: "fe4e2bf389fe05dab6390564e40012d8d36eca19"
Location: http://localhost:8080/rest/parent/container/<autogenerated-ds-id>

Body:
http://localhost:8080/rest/parent/container/<autogenerated-ds-id>

Status:

- **201 Created**
- **404 Not Found** (resource does not exist – use PUT to specify a new resource name)
- **409 Conflict** (checksum mismatch)

PUT

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

For **RDFSource** resources

Some resource properties are repository managed and cannot be removed or modified. Any attempt to do so may result in a 4xx error.

Request Headers:

- **CONTENT-DISPOSITION** (Optional) The filename provided in the content disposition header will be stored in a ebucore:filename property. (See 'Example (6): Uploaded file with filename' in POST above)
- **CONTENT-TYPE** (Optional) MIME type of the uploaded binary or RDF content

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.

If the MIME type corresponds to a supported **RDF format or SPARQL-Update**, and you also provide a link: <http://www.w3.org/ns/ldp#NonRDFSource>; rel="type" header. Then the contents will be stored as a binary resource with the provided MIME type as Content-type.

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

**Supported RDF formats**: text/turtle, text/rdf+n3, application/n3, text/n3, application/rdf+xml, application/n-triples, text/html, text/plain, application/json

- **IF-MATCH**
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. Valid 'Digest' algorithms/values are: sha, sha-256, or md5.

With the value handling=lenient; received="minimal" allows replacing the properties of a container without having to provide all of the server-managed triples.

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
curl -X PUT -H "Content-Type: text/turtle" --data-binary @new-triples.rdf "http://localhost:8080/rest/node/to/update"

Request Body:

```turtle
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: "ef214795c3b9109389fa542a6c081976e1e9587"*
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
curl -X PUT --upload-file image.jpg -H"Content-Type: image/jpeg" -H"digest: sha=cbl5766f22e8e3e11d611b616e3e2f5ce9bd941" "http://localhost:8080/rest/new/image"
```

Response:
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**

```bash
curl -X PUT -H "Link: <http://www.example.com/file>; rel="http://fedora.info/definitions/fcrepo#ExternalContent"; handling="redirect"; type="text/plain"" "http://localhost:8080/rest/node/to/create"
```

Response:

Status: 201 Created

Headers:

ETag: "ef214795c3b9109389ffa542a6c081976e1e9587"
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

---

**PATCH**

Modify the triples associated with a resource with SPARQL-Update

**Request Headers:**

- IF-MATCH
- IF-UNMODIFIED-SINCE
- CONTENT-TYPE

**Example (1):**
curl -X PATCH -H "Content-Type: application/sparql-update" --data-binary @body.rdf" "http://localhost:8080/rest/node/to/update"

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 of a binary’s description:

curl -X PATCH -H "Content-Type: application/sparql-update" --data-binary @body.rdf" "http://localhost:8080/rest/binary/resource/to/binary/fcr:metadata"

Request Body:

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

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

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
Last-Modified: Thu, 20 Nov 2014 15:44:32 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
```

**MOVE**

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

**NOTE:** The MOVE action has been deprecated and will be removed in a future version of Fedora.

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:**

- 200 OK
- 201 Created
- 409 Source path doesn't exist
- 412 Destination path already exists
Destination URI isn't a valid resource path

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

NOTE: The COPY action has been deprecated and will be removed in a future version of Fedora.

Example:

```
```

Response:

Status: 201 Created

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

**Status:**

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

**Versioning**

**RESTful HTTP API - Versioning**

- Working with a versionable resource (a LDPRv)
  - GET Retrieve the versioned resource from a specific point in time.
- Working with a versions container (a LDPCv)
  - GET Get a list of the available versions of an object
  - POST Create a new versioned resource (a new LDPRm)
  - DELETE Remove the versions container and all versioned resources
- Working with a versioned resource (a LDPRm)
  - GET Get a specific versioned resource
  - DELETE Remove a previous version of an object

**Services**

**Backup and Restore**

**RESTful HTTP API - Backup and Restore**

Request URI: /rest/fcr:backup
Methods: 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

Example:

curl -X POST -d /path/to/backup/destination "http://localhost:8080/rest/fcr:backup"

Response:

Status: 200 OK

Request URI: /rest/fcr:restore

Methods: POST

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

Note: Restoring a backup replaces the repository content with the contents of the backup, so any data in the repository will be lost.

Example:

curl -X POST -d "/tmp/fcrepo4-data/path/to/backup/directory" "http://localhost:8080/rest/fcr:restore"

Response:
Fixity

RESTful HTTP API - Fixity

Request URI: /path/to/some(binary

Methods: HEAD, GET

**HEAD** Request the fixity checksum for a given digest algorithm

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

**Request Headers:**

```
WANT-DIGEST: md5, sha, sha-256
```

**Example:**

```
curl -I -H "Want-Digest: sha-256" "http://localhost:8080/rest/path/to/some(binary"
```

**Response (fixity success):**
Status: 200 OK

Headers:
ETag: "51c5ed5ffe4b6c79233cc7573a387920f7d171e7"
Last-Modified: Fri, 04 May 2018 18:14:47 GMT
Content-Type: image/jpeg
Accept-Ranges: bytes
Content-Disposition: attachment; filename="IMG_4023.JPG.jpg"; creation-date="Fri, 04 May 2018 18:14:47 GMT"; modification-date="Fri, 04 May 2018 18:14:47 GMT"; size=47021
Link: <http://www.w3.org/ns/ldp#Resource>;rel="type"
Link: <http://www.w3.org/ns/ldp#NonRDFSource>;rel="type"
Link: <http://localhost:8080/rest/examples/binary/fcr:acl>; rel="acl"
Link: <http://localhost:8080/rest/examples/binary/fcr:metadata>; rel="describedby"
Link: <http://localhost:8080/static/constraints/NonRDFSourceConstraints.rdf>; rel="http://www.w3.org/ns/ldp#constrainedBy"
Link: <http://localhost:8080/rest/examples/binary>; rel="timegate"
Link: <http://localhost:8080/rest/examples/binary>; rel="original"
Link: <http://localhost:8080/rest/examples/binary/fcr:versions>; rel="timemap"
Link: <http://mementoweb.org/ns#OriginalResource>; rel="type"
Accept-External-Content-Handling: copy, redirect, proxy
Allow: DELETE, HEAD, GET, PUT, OPTIONS
Digest: sha-256=08692ca74dc1e08c84f90fed5e64e72617be850a7c7adca88b03f6c3d3db406
Content-Length: 47021

Status:
200 OK
400 Bad request (the 'Want-Digest' request-header algorithm type is either invalid or unsupported)
404 Resource not found

NOTE: While the following feature is still available in Fedora 5, it is deprecated and may be removed in a future release.

Request URI: /path/to/some/resource/fcr:fixity
Methods: GET

GET Get the fixity report for an object
Checking fixity requires retrieving the content from the binary store and may take some time.
The /fcr:fixity endpoint is not part of the Fedora API Specification. This feature is currently supported for backwards compatibility reasons and may be removed in a future release.

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:
Note: Default Fixity Algorithm

When ingesting a binary resource, one or more checksums may be provided, see API reference (POST example 4, POST example 4b, PUT example 3). The supported algorithms are: SHA-1, SHA-256 and MD5.

By default, the algorithm used by the /fcr:fixity endpoint is SHA-1. However, that may be changed - per resource - to one of the other supported algorithms by configuring the property: fedoraconfig:defaultDigestAlgorithm. See API reference (PATCH example 2).
RESTful HTTP API - Transactions


Methods: GET, POST

### GET

Get the current status of the repository in a transaction

**Example:**

```bash
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.

**Status:**

- **200 OK:** If the request was successful
- **410 Gone:** Transaction not found

### 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:**

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

**Response:**

- **Status:** 201 Created
- **Headers:**
  - Location: http://localhost:8080/rest/tx:83e34464-144e-43d9-af13-b3464a1fb9b5
  - Expires: Sat, 16 Nov 2013 00:32:57 GMT

**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.

```bash
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"
```

**Status:**
**POST** Keep an existing transaction alive

**Example:**


**Response:**

Status: 204 No Content

Headers:
Location: http://localhost:8080/rest/tx:83e34464-144e-43d9-af13-b3464a1fb9b5
Expires: Sat, 16 Nov 2013 03:35:43 GMT

**Status:**

- **204** No Content: if the transaction is renewed successfully
- **410** Gone: if the transaction doesn't exist

**POST** 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

**POST** Rollback and close an open transaction

**Example:**


**Response:**
<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>No Content: if the transaction is discarded successfully</td>
</tr>
<tr>
<td>410</td>
<td>Gone: if the transaction doesn't exist</td>
</tr>
</tbody>
</table>