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/n3+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](https://www.w3.org/TR/ldp/), 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

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

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
- 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

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

Response:
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,PATCH,OPTIONS
Accept-Post: text/turtle,text/rdf+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:

<?xml version="1.0" encoding="UTF-8"?>
<http://fedora.info/definitions/v4/repository#hasParent> <http://localhost:8080/rest/path/to> ;
a <http://www.w3.org/2002/07/op/containers> [ http://www.w3.org/2002/07/op/containers#DirectContainer ];
<http://www.w3.org/2002/07/op/containers#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 (3): Prefer headers

```bash
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:
Create new resources within a LDP container

Query Parameters:

- **CHECKSUM** (Optional) Provide a SHA-1 checksum which will be checked against the uploaded content to ensure error-free transfer.

Request Headers:

- **CONTENT-DISPOSITION** (Optional) The filename provided in the content disposition header will be stored in a `premis:hasOriginalName` property.
- **CONTENT-TYPE** (Optional) MIME type of the uploaded binary or RDF content.
- **CONTENT-LOCATION** (Optional) A URI to a resource to use instead of the request body

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.

- **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.

Example (1): Create a new child node

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

Response:
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
Headers:
ETag: "3ac31b09de1536a332d9c6ee8453669947721840"
Location: http://localhost:8080/rest/some/path/to/a/new/resource

Body:
http://localhost:8080/rest/some/path/to/a/new/resource

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

```bash
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: "487f188240d2be3ae32d49958c0e1e1a8224be0a"
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 (4): Uploaded file with checksum

```bash
curl -i -X POST --data-binary @picture.jpg" http://localhost:8080/rest/parent/container?checksum=urn:sha1:cb1a576f22e8e3e110611b616e3e2f5ce9dbb941"
```

Response:

### Example (5): Uploaded file with checksum mismatch

```bash
```

**Response:**

<table>
<thead>
<tr>
<th>Status</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:**

- **CONTENT-TYPE**
  - 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
- **IF-MATCH**
- **IF-UNMODIFIED-SINCE**
- **CONTENT-LOCATION** (Optional) A URI to a resource to use instead of the request body

**Query Parameters:**

- **CHECKSUM**
  - Checksum of the binary content (in the form `urn:sha1:<sha1-hash>`). If the checksum doesn't match, a 409 Conflict will be returned.
Example (1): Updating properties with RDF content

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

```sh
```

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> ;
```

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

```sh
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> ;
<> dc:title "some-resource-title".
```

Response:

```
Status: 204 No Content
```

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

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

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

```bash
curl -X PUT --upload-file image.jpg -H"Content-Type: image/jpeg" "http://localhost:8080/rest/new/image"
```

Response:

```
Status: 201 Created
Response Headers:
ETag: "ef214795c3b9109389ffa542a6c081976e1e9587"
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
```

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:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>No Content</td>
</tr>
<tr>
<td>412</td>
<td>Precondition Failed</td>
</tr>
</tbody>
</table>

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

```bash
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
Response Headers:
ETag: "ef214795c3b9109389ffa542a6c081976e1e9587"
Last-Modified: Mon, 19 May 2014 20:52:57 GMT
Location: http://localhost:8080/rest/node/to/create
Link: <http://localhost:8080/rest/node/to/create/fcr:metadata>; rel="describedby"
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-LOCATION (Optional) A URI to a resource to use instead of the request body

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): Updating multiple containers with a single SPARQL Update

```
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
```

Status:

- **204** No Content
- **412** Precondition Failed

DELETE Delete a resource

Example:

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

Response:
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:

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:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>No Content</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
</tr>
</tbody>
</table>

```
HEAD Retrieve the resource headers
```

Example:

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

Response:
**Status:** 200 OK

- **ETag:** "bbdd92e395800153a686773f773b3c80a51f47b"
- **Last-Modified:** Wed, 28 May 2014 18:31:36 GMT
- **Link:** 
  - `<http://www.w3.org/ns/ldp#Resource>;rel="type"`
  - `<http://www.w3.org/ns/ldp#Container>;rel="type"`
  - `<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:** 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
```

<table>
<thead>
<tr>
<th>Status</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Created</td>
</tr>
<tr>
<td>409</td>
<td>Source path doesn't exist</td>
</tr>
<tr>
<td>412</td>
<td>Destination path already exists</td>
</tr>
<tr>
<td>502</td>
<td>Destination URI isn't a valid resource path</td>
</tr>
</tbody>
</table>

**Copy**

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
```

<table>
<thead>
<tr>
<th>Status</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Created</td>
</tr>
<tr>
<td>409</td>
<td>Source path doesn't exist</td>
</tr>
<tr>
<td>412</td>
<td>Destination path already exists</td>
</tr>
<tr>
<td>502</td>
<td>Destination URI isn't a valid resource path</td>
</tr>
</tbody>
</table>

**Export and Import**

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

Methods: GET, POST

⚠️ Exporting large trees of resources may fail because of memory limitations. Exporting 250,000 simple objects (only the system-defined properties) was successful and resulted in a 400MB export file, but exporting larger sets of resources failed. To work around this limitation, you can segment your repository (for example, into several top-level directories each containing 250,000 or fewer resources) and export each top-level directory as a separate export file.
GET fcr:export Retrieve serialized form of a resource. By default, no subtree and binary content will be exported

Request Headers:

- FORMAT: Export formats; the exact formats available depend on the repository configuration.

Parameters:

- recurse: To export subtree with its value set to true. For example, recurse=true. Default is false.
- skipBinary: To export binary content with its value set to false. For example, skipBinary=false. Default is true.

The export formats that are available are shown as properties on resources, e.g. this resource exports jcr/xml:

```xml
<dc:format rdf:resource="#export"/>
```

Example:

To export a resource with no subtree and no binary content:

```bash
curl "http://localhost:8080/rest/path/to/some/resource/fcr:export"
```

To export a resource with subtree and binary content:

```bash
```

Response:

- Status: 200
- OK: if the resource is exported successfully
- 404: Not Found: if the resource does not exist
**POST**  
**fcr:import** Import a serialized resource

**Request Headers:**

- **FORMAT** Import formats. The exact formats available depend on the repository configuration.
- **CONTENT-LOCATION** (Optional) A URI to a resource to use instead of the request body

**Example:**

```bash
curl -X POST --data-binary @export.xml "http://localhost:8080/rest/path/to/some/fcr:import?format=jcr/xml"
```

**Body:**

```
```

**Response:**

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

**Status:**

- **201** No Content: if the resource is imported successfully
- **404** Not Found: if the path does not exist
- **409** Conflict: if a node with the same UUID exists

**Response:**

```
Item already exists
```

**Versioning**

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

**Methods:** GET, POST, PATCH

**GET**  
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:**

```bash
```
### Status:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>404</td>
<td>Not Found: This resource is not versioned or this resource is not exist.</td>
</tr>
</tbody>
</table>

**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/rdf+xml, application/x-turtle, application/xhtml+xml, application/xml, text/html, text/n3, text/plain, text/rdf+n3, text/turtle

**Example:**

```
```

**Response:**
The requested resource is not available.

POST Create a new version of an object

Request Headers

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

Example:

```
```

Response:

Status: 204 No Content
409 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
```

For previous versions for which explicit names were not provided, you may use the UUID property which you can find by parsing the response from the listing of available versions.

Response:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>No Content: if the version is reverted successfully</td>
</tr>
<tr>
<td>404</td>
<td>Not Found: if the version does not exist</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>No Content</td>
</tr>
</tbody>
</table>

Warning: 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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>No Content: if the version is reverted successfully</td>
</tr>
<tr>
<td>400</td>
<td>Bad Request: if trying to delete the most recent version</td>
</tr>
<tr>
<td>404</td>
<td>Not Found: if the version does not exist</td>
</tr>
</tbody>
</table>

Response: Cannot remove current version

Response: The requested resource is not available.
Services

Access Roles

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

Methods: GET, POST, DELETE

⚠️ Arbitrary access roles may be set through this API. However, they are only enforced when paired with a Policy Enforcement Point that is aware of access roles.

**GET**

Get a list of all the roles assigned to principals on this resource

Output formats: application/json

Query Parameters:

- **EFFECTIVE**: When supplied, gets the list of effective roles for this resource, including those inherited from the parent

Example:

```
curl "http://localhost:8080/rest/49/3d/24/41/493d2441-0541-41c7-a23b-09d1f17d4a0f/fcr:accessroles"
```

Response:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK: if access roles retrieved successfully</td>
</tr>
<tr>
<td>204</td>
<td>No Content: if access roles not assigned</td>
</tr>
<tr>
<td>404</td>
<td>Not Found: if the resource doesn’t exist</td>
</tr>
</tbody>
</table>

**POST**

Set the roles assigned on this resource

Request Headers:

- **CONTENT-TYPE**: application/json

Example:
curl -X POST -H "Content-Type: application/json" --data-binary @post.txt "http://localhost:8080/rest/49/3d/24/41/493d2441-0541-41c7-a23b-09d1f17d4a0f/fcr:accessroles"

Body:
{
  "principal name 1" : [ "reader" ],
  "principal name 2" : [ "writer" ],
  "principal name 3" : [ "admin" ]
}

Response:

Status: 201 Created
Headers:
Location: http://localhost:8080/rest/49/3d/24/41/493d2441-0541-41c7-a23b-09d1f17d4a0f/fcr:accessroles
Content-Length: 0
Server: Jetty(8.1.11.v20130520)

Status:

201 Created: if the access roles were created successfully
400 Bad Request: if there was an error processing the access roles declaration
415 Unsupported Media Type: if the Content-Type of the uploaded content isn't application/json

DELETE Clear all roles assignments from this resource

Example:

curl -X DELETE "http://localhost:8080/rest/49/3d/24/41/493d2441-0541-41c7-a23b-09d1f17d4a0f/fcr:accessroles"

Response:

Status: 204 No Content
Headers:
Server: Jetty(8.1.11.v20130520)

Status:

204 No Content: if the access roles are deleted successfully
404 Not Found: if the resource doesn't exist

Backup and Restore

Request URI: /
Methods: GET, POST

POST fcr:backup Initiate a (consistent) backup of the repository
The backup will be stored in a temporary directory on the server. 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
```

**POST**

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

Example:

```
```

Response:

```
Status: 204 No Content
```

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>204</td>
<td>Backup restored</td>
</tr>
<tr>
<td>500</td>
<td>Error restoring backup</td>
</tr>
</tbody>
</table>

**Fixity**

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.
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
Headers:
Content-Type: text/turtle

Body:
```html


<http://localhost:8080/rest/path/to/some/resource#fixity/1400589459772> <http://www.loc.gov/premis/rdf/v1#hasContentLocation> <info://org.modeshape.jcr.value.binary.FileSystemBinaryStore@7bcc39fb/fcrepo4/fcrepo-webapp/fcrepo4-data/fcrepo.binary-store-path#f7d787ee7fc58ce7fc257ae0667a2c65476be750> .

<info://org.modeshape.jcr.value.binary.FileSystemBinaryStore@7bcc39fb/fcrepo4/fcrepo-webapp/fcrepo4-data/fcrepo.binary-store-path#f7d787ee7fc58ce7fc257ae0667a2c65476be750> a <http://www.loc.gov/premis/rdf/v1#ContentLocation> ;
  <http://www.loc.gov/premis/rdf/v1#hasContentLocationValue> "info://org.modeshape.jcr.value.binary.FileSystemBinaryStore@7bcc39fb/fcrepo4/fcrepo-webapp/fcrepo4-data/fcrepo.binary-store-path#f7d787ee7fc58ce7fc257ae0667a2c65476be750"^^<http://www.w3.org/2001/XMLSchema#string> .
```

Response (fixity failure):

```
Status: 200 OK
Headers:
Content-Type: text/turtle

Body:
```html


<http://localhost:8080/rest/path/to/some/resource#fixity/1400589459772> <http://www.loc.gov/premis/rdf/v1#hasContentLocation> <info://org.modeshape.jcr.value.binary.FileSystemBinaryStore@7bcc39fb/fcrepo4/fcrepo-webapp/fcrepo4-data/fcrepo.binary-store-path#f7d787ee7fc58ce7fc257ae0667a2c65476be750> .

  <http://www.loc.gov/premis/rdf/v1#hasMessageDigest> <urn:sha1:b04bded0d83b74ac0c700945e24e3e823eb5821> ;
  <http://www.loc.gov/premis/rdf/v1#hasSize> "1324943"^^<http://www.w3.org/2001/XMLSchema#int> .
```
Node Types

Request URI: /fcr:nodetypes
Methods: GET

GET get a list of registered types (as RDFS triples)

Request
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:
curl "http://localhost:8080/rest/fcr:nodetypes"

Response:

Status: 200 OK
Headers:
Content-Type: text/turtle
Body:

<http://fedora.info/definitions/v4/rest-api#object>
  a <http://www.w3.org/2000/01/rdf-schema#Class> ;
  <http://www.w3.org/2000/01/rdf-schema#label>
    "fedora:object" ;
  <http://www.w3.org/2000/01/rdf-schema#subClassOf>
    <http://fedora.info/definitions/v4/rest-api#resource> .

<http://fedora.info/definitions/v4/rest-api#resource>
  a <http://www.w3.org/2000/01/rdf-schema#Class> ;
  <http://www.w3.org/2000/01/rdf-schema#label>
    "fedora:resource" ;
  <http://www.w3.org/2000/01/rdf-schema#subClassOf>
    <http://fedora.info/definitions/v4/rest-api#relations> ,
    <http://www.jcp.org/jcr/mix/1.0created> ,
    <http://www.jcp.org/jcr/mix/1.0lastModified> ,
    <http://www.jcp.org/jcr/mix/1.0lockable> ,
    <http://www.jcp.org/jcr/mix/1.0versionable> .

[...]
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:{txid}/fcr:tx

Example:

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.

```
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:

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

**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:

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

**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:

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

**Transform**

Request URI: `/fcr:transform/{program}`

Methods: GET, POST

**GET** get a resource transformed with the default transform

Request Headers:

Example:

```
@prefix fedora : <http://fedora.info/definitions/v4/repository#>
id = . :: xsd:string ;
title = dc:title :: xsd:string ;
uuid = fedora:uuid :: xsd:string ;
```

```
curl "http://localhost:8080/rest/49/3d/24/41/493d2441-0541-41c7-a23b-09d1f17d4a0f/fcr:transform/default"
```

Response:

<table>
<thead>
<tr>
<th>Status</th>
<th>Headers</th>
<th>Body</th>
</tr>
</thead>
</table>
| 200 OK | Content-Type: application/json, Transfer-Encoding: chunked, Server: Jetty(8.1.11.v20130520) | ```

```
**OK**: if the transform is applied successfully

**400**: Bad Request: if the program doesn’t exist

**404**: Not Found: if the resource doesn’t exist

Note

To reference a stored “program” in the GET form of the fcr:transform service, the LDPath transformation directive must be stored at the following path:

```
```

Where `<program-name>` can be any name, and `<resource-type>` is the type of Fedora resource on which the transform will be applied:

- "fedora:Container"
- "fedora:nonRdfSourceDescription"
- "fedora:Resource"

**Example**:

1. Store a new program.

   ```
   ```

**Response**

Status 201 Created

Headers:
Content-Type: text/plain


2. Get a resource transformed with the stored transform

   ```
   ```

**Response**

Status 200 OK

Header:
Server: Apache-Coyote/1.1
Content-Type: application/json

Body:
```
[
  {
    "id": ["http://localhost:8080/rest/pid5"],
    "title": ["some-resource-title"],
    "description": [],
    "uuid": ["caa7bc6c-b80b-4e30-8ec1-15e90937e3be"]
  }
]
```

**POST**

get a resource transformed with the supplied transform

Output formats: application/json,text/tab-separated-values,text/csv,text/sse,text/plain,application/sparql-results+json,application/sparql-results+xml,application/sparql-results+bio,text/turtle,text/rdf+n3,application/N-triples.application/rdf_xml

**Example**:

```
curl -X POST -H "Content-Type: application/rdf+ldpath" --data-binary @post.txt "http://localhost:8080/rest/49/3d/24/41/493d2441-0541-41c7-a23b-09d1f17d4a0f/fcr:transform"

Body:
@prefix fcrepo : <http://fedora.info/definitions/v4/repository#>
  id = . :: xsd:string ;
  title = dc:title :: xsd:string ;
  uuid = fcrepo:uuid :: xsd:string ;

Response:

Status: 200 OK

Headers:
Content-Type: application/json
Transfer-Encoding: chunked
Server: Jetty(8.1.11.v20130520)

Body:
[
  {
    "id": "http://localhost:8080/rest/49/3d/24/41/493d2441-0541-41c7-a23b-09d1f17d4a0f",
    "title": [],
    "uuid": "07630a24-5a0b-4ba7-80ab-0691f68667ce"
  }
]

Status:
- 200 OK: if the transform is applied successfully
- 400 Bad Request: if there was an error parsing or processing the transform
- 404 Not Found: if the resources doesn't exist