SPARQL Recipes

This page provides detailed steps for creating repository content via the REST API, and SPARQL queries to demonstrate that content has been synchronized to an external triplestore. For more details on setting up triplestore sync, see External Triplestore. Much more information about the REST API is available at: RESTful HTTP API and Another Take on the REST API .

1. Finding Containers by Binary Name or MIME Type

Create a few sample containers with either a PDF or a text file, or both attached as binaries (the following commands assume you have two files named test.pdf and test.txt in your current working directory; please substitute working filenames as needed):

File: container-indexing.rdf

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX indexing: <http://fedora.info/definitions/v4/indexing#>
DELETE { }
INSERT {
  <> indexing:hasIndexingTransformation "default";
  rdf:type indexing:Indexable }
WHERE { }
```

1a. Creating Sample Containers

```
curl -X PUT "http://localhost:8080/rest/objects/101"
curl -X PATCH -H "Content-type: application/sparql-update" --data-binary "@container-indexing.rdf"
"http://localhost:8080/rest/objects/101"
curl -X PUT -H "Content-type: application/pdf" --data-binary "@test.pdf" "http://localhost:8080/rest/objects/101
/master"
curl -X PATCH -H "Content-type: application/sparql-update" --data-binary "@container-indexing.rdf"
"http://localhost:8080/rest/objects/101/master/fcr:metadata"
curl -X PUT "http://localhost:8080/rest/objects/102"
curl -X PATCH -H "Content-type: application/sparql-update" --data-binary "@container-indexing.rdf"
"http://localhost:8080/rest/objects/102"
curl -X PUT -H "Content-type: text/plain" --data-binary "@test.txt" "http://localhost:8080/rest/objects/102
/master"
curl -X PATCH -H "Content-type: application/sparql-update" --data-binary "@container-indexing.rdf"
"http://localhost:8080/rest/objects/102/master/fcr:metadata"
curl -X PUT "http://localhost:8080/rest/objects/103"
curl -X PATCH -H "Content-type: application/sparql-update" --data-binary "@container-indexing.rdf"
"http://localhost:8080/rest/objects/103"
curl -X PUT -H "Content-type: application/pdf" --data-binary "@test.pdf" "http://localhost:8080/rest/objects/103
/master"
curl -X PUT -H "Content-type: text/plain" --data-binary "@test.txt" "http://localhost:8080/rest/objects/103
/text"
curl -X PATCH -H "Content-type: application/sparql-update" --data-binary "@container-indexing.rdf"
"http://localhost:8080/rest/objects/103/master/fcr:metadata"
curl -X PATCH -H "Content-type: application/sparql-update" --data-binary "@container-indexing.rdf"
"http://localhost:8080/rest/objects/103/text/fcr:metadata"
```

Find containers with a binary named "text":

1b. Selecting Containers With Binaries Named "text"

```
prefix fcrepo: <http://fedora.info/definitions/v4/repository#>
select ?container where {
    ?ds fcrepo:mixinTypes "fedora:NonRdfSourceDescription" .
    ?ds fcrepo:hasParent ?container .
    filter(str(?ds)=concat(str(?container),'/text/fcr:metadata'))
}
```

Expected Results

container
<http://localhost:8080/rest/objects/103>

Find containers with a PDF binary:

1c. Selecting Containers With PDF Binaries

```
prefix fcrepo: <http://fedora.info/definitions/v4/repository#>
prefix relation: <http://www.iana.org/assignments/relation/>
select ?container where {
    ?ds fcrepo:mixinTypes "fedora:NonRdfSourceDescription" .
    ?ds fcrepo:hasParent ?container .
    ?ds relation:describes ?content .
    ?content fcrepo:mimeType "application/pdf"
}
```

Expected Results

container
<http://localhost:8080/rest/objects/101>
<http://localhost:8080/rest/objects/103>

2. Attach metadata properties directly to containers

Create an container and attach the dc:title property:

2a. Creating Sample Containers

```
curl -X PUT http://localhost:8080/rest/objects/201
echo "prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> prefix index: <http://fedora.info/definitions/v4
/indexing#>  insert data { <> rdf:type index:Indexable . }" \
| curl -X PATCH --upload-file - http://localhost:8080/rest/objects/201
echo "prefix dc: <http://purl.org/dc/elements/1.1/> insert data { <http://localhost:8080/rest/objects/201> dc:
title 'Foo' . }" \
| curl -X PATCH --upload-file - http://localhost:8080/rest/objects/201
```

Find containers with titles:

2b. Selecting Containers With Titles

```
prefix dc: <http://purl.org/dc/elements/1.1/>
select ?object ?title where { ?object dc:title ?title }
```

Expected Results

container	title
<http://localhost:8080/rest/objects/201>	"Foo"

3. Find Containers By Collection Membership

Create three containers and three collections:

3a. Creating Sample Containers And Collections

```
curl -X PUT http://localhost:8080/rest/objects/col1
curl -X PUT http://localhost:8080/rest/objects/col2
curl -X PUT http://localhost:8080/rest/objects/col3
curl -X PUT http://localhost:8080/rest/objects/obj1
curl -X PUT http://localhost:8080/rest/objects/obj2
curl -X PUT http://localhost:8080/rest/objects/obj3
```

Mark the containers Indexable:

3b. Marking containers Indexable

```
echo "insert data { <> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://fedora.info/definitions/v4
/indexing#Indexable> . }" \
| curl -X PATCH --upload-file - http://localhost:8080/rest/objects/obj1
echo "insert data { <> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://fedora.info/definitions/v4
/indexing#Indexable> . }" \
| curl -X PATCH --upload-file - http://localhost:8080/rest/objects/obj2
echo "insert data { <> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://fedora.info/definitions/v4
/indexing#Indexable> . }" \
| curl -X PATCH --upload-file - http://localhost:8080/rest/objects/obj3
echo "insert data { <> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://fedora.info/definitions/v4
/indexing#Indexable> . }" \
| curl -X PATCH --upload-file - http://localhost:8080/rest/objects/col1
echo "insert data { <> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://fedora.info/definitions/v4
/indexing#Indexable> . }" \
| curl -X PATCH --upload-file - http://localhost:8080/rest/objects/col2
echo "insert data { <> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://fedora.info/definitions/v4
/indexing#Indexable> . }" \
| curl -X PATCH --upload-file - http://localhost:8080/rest/objects/col3
```

Link each container to a collection:

3c. Linking Containers To Collections

```
echo 'insert data { <http://localhost:8080/rest/objects/obj1> <http://some-vocabulary.org/rels-
ext#isMemberOfCollection> <http://localhost:8080/rest/objects/col1> . }' | curl -X PATCH --upload-file -
http://localhost:8080/rest/objects/obj1
echo 'insert data { <http://localhost:8080/rest/objects/obj2> <http://some-vocabulary.org/rels-
ext#isMemberOfCollection> <http://localhost:8080/rest/objects/col2> . }' | curl -X PATCH --upload-file -
http://localhost:8080/rest/objects/obj2
echo 'insert data { <http://localhost:8080/rest/objects/obj3> <http://some-vocabulary.org/rels-
ext#isMemberOfCollection> <http://localhost:8080/rest/objects/col3> . }' | curl -X PATCH --upload-file -
http://localhost:8080/rest/objects/obj3
```

Link the collections in a hierarchy:

3d. Linking Collections In A Hierarchy

```
echo 'insert data { <http://localhost:8080/rest/objects/col1> <http://some-vocabulary.org/rels-ext#hasPart>
<http://localhost:8080/rest/objects/col2> . }' | curl -X PATCH --upload-file - http://localhost:8080/rest
/objects/col1
echo 'insert data { <http://localhost:8080/rest/objects/col2> <http://some-vocabulary.org/rels-ext#hasPart>
<http://localhost:8080/rest/objects/col3> . }' | curl -X PATCH --upload-file - http://localhost:8080/rest
/objects/col2
```

Find containers directly attached to a collection:

3e. Selecting Containers Directly Linked To A Collection

```
select ?obj ?col where { ?obj <http://some-vocabulary.org/rels-ext#isMemberOfCollection> ?col }
```

Expected Results

obj	col
<http://localhost:8080/rest/objects/obj1>	<http://localhost:8080/rest/objects/col1>
<http://localhost:8080/rest/objects/obj2>	<http://localhost:8080/rest/objects/col2>
<http://localhost:8080/rest/objects/obj3>	<http://localhost:8080/rest/objects/col3>

Find containers directly or indirectly attached (recursive retrieval along the rels:hasPart chain). Applications developed using Fedora 3 may use Mulgara's walk() function for queries that navigate a hierarchy. That functionality is now part of standard SPARQL 1.1 (Jena's property paths documentation provides some good examples).

3f. Select Containers Directly Or Indirectly Linked To A Collection

```
prefix rels: <http://some-vocabulary.org/rels-ext#>
select ?obj where {
  <http://localhost:8080/rest/objects/col1> rels:hasPart* ?col
  . ?obj rels:isMemberOfCollection ?col
}
```

Expected Results

containers
<http://localhost:8080/rest/objects/obj1>
<http://localhost:8080/rest/objects/obj2>
<http://localhost:8080/rest/objects/obj3>

Also link a container to a project:

3g. Link Containers To A Project

```
curl -X PUT http://localhost:8080/rest/objects/proj1
echo "insert data { <> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://fedora.info/definitions/v4
/indexing#Indexable> . }" \
| curl -X PATCH --upload-file - http://localhost:8080/rest/objects/proj1
echo "prefix ex: <http://example.org/> insert data { <http://localhost:8080/rest/objects/obj1> ex:project
<http://localhost:8080/rest/objects/proj1> . }" \
| curl -X PATCH --upload-file - http://localhost:8080/rest/objects/obj1
```

Find containers in a collection or linked to a project:

3h. Select Containers Linked To A Collection or A Project

```
prefix rels: <http://some-vocabulary.org/rels-ext#>
prefix ex: <http://example.org/>
select ?obj where {
  { ?obj ex:project <http://localhost:8080/rest/objects/proj1> }
  UNION
  { ?obj rels:isMemberOfCollection <http://localhost:8080/rest/objects/col2> }
}
```

Expected Results

obj
<http://localhost:8080/rest/objects/obj1>
<http://localhost:8080/rest/objects/obj2>

Count containers instead of listing them:

3i. Count the Containers Linked to a Collection or a Project

```
prefix rels: <http://some-vocabulary.org/rels-ext#>
prefix ex: <http://example.org/>
select (count(distinct ?obj) as ?count) where {
  { ?obj ex:project <http://localhost:8080/rest/objects/proj1> }
  UNION
  { ?obj rels:isMemberOfCollection <http://localhost:8080/rest/objects/col2> }
}
```

Expected Results

count
"2"^^<http://www.w3.org/2001/XMLSchema#integer>