

Quick Start with WebAC

In this quick start, you will use a Fedora 4 server with the WebAC Authorization module enabled to create a sample resource and an ACL for that resource, verify that access to that resource is correctly restricted, and finally modify the ACL to allow you to update the resource.

Prerequisites

- Fedora 4 with WebAC module enabled (you can use one of the [pre-built WAR files](#) from the fcrepo-webapp-plus project)
- curl

The commands in this guide assume that your Fedora 4 is running at <http://localhost:8080/fcrepo>.

Steps

Create these three files:

acl.ttl

```
@prefix webac: <http://fedora.info/definitions/v4/webac#>.

<> a webac:Acl .
```

foo.ttl

```
@prefix acl: <http://www.w3.org/ns/auth/acl#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.

<> acl:accessControl </fcrepo/rest/acl>;
    dc:title "Hello, World!".
```

authz.ttl

```
@prefix acl: <http://www.w3.org/ns/auth/acl#>.

<> a acl:Authorization;
    acl:accessTo </fcrepo/rest/foo>;
    acl:agent "user1";
    acl:mode acl:Read.
```

Upload these files into the repository:

```
$ curl -X PUT http://localhost:8080/fcrepo/rest/acl -u admin1:password3 \
    -H "Content-Type: text/turtle" --data-binary @acl.ttl
$ curl -X PUT http://localhost:8080/fcrepo/rest/foo -u admin1:password3 \
    -H "Content-Type: text/turtle" --data-binary @foo.ttl
$ curl -X PUT http://localhost:8080/fcrepo/rest/acl/authz -u admin1:password3 \
    -H "Content-Type: text/turtle" --data-binary @authz.ttl
```

(Note: The order you upload these in is important, since foo references acl, and authz references foo.)

Now user1 is able to read the resource at <http://localhost:8080/rest/foo>, but user2 cannot. To test this, try the following two commands:

```
$ curl -i http://localhost:8080/fcrepo/rest/foo -u user1:password1
$ curl -i http://localhost:8080/fcrepo/rest/foo -u user2:password2
```

The first request should succeed with a **200 OK** response code, and the second should fail with a **403 Forbidden**.

To demonstrate that user1 indeed only has read-only access to foo, we can try updating foo. Create a file named **foo.sparql** with the following contents:

foo.sparql

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

INSERT {
  <> dc:description "Quick Start with WebAC and Fedora 4".
}
WHERE {}
```

Then run this to attempt to update foo:

```
$ curl -i -X PATCH http://localhost:8080/fcrepo/rest/foo -u user1:password1 \
  -H "Content-Type: application/sparql-update" \
  --data-binary @foo.sparql
```

This request should fail with a **403 Forbidden** response, since user1 has read-only access to foo. To add write access for user1, we will need to update the acl/authz resource as admin. Create a file named **authz.sparql** with the following contents:

authz.sparql

```
PREFIX acl: <http://www.w3.org/ns/auth/acl#>

INSERT {
  <> acl:mode acl:Write .
}
WHERE {}
```

Run this command to update the ACL authorization:

```
$ curl -i -X PATCH http://localhost:8080/fcrepo/rest/acl/authz -u admin1:password3 \
  -H "Content-Type: application/sparql-update" \
  --data-binary @authz.sparql
```

If the update to the authorization was successful, you will see a **204 No Content** response.

Now you should be able to re-run the earlier command to update the foo resource as user1:

```
$ curl -i -X PATCH http://localhost:8080/fcrepo/rest/foo -u user1:password1 \
  -H "Content-Type: application/sparql-update" \
  --data-binary @foo.sparql
```

Now this should return a **204 No Content** response. To verify that the update happened, you can also go to <http://localhost:8080/fcrepo/rest/foo> in your web browser, and confirm that it has both **dc:title** and **dc:description** properties.

Access Control Link Header

 When you perform a successful **GET** request on a resource that has an ACL associated with it (or with an ancestor), you will receive an additional header of the format.

```
Link: <http://localhost:8080/fcrepo/rest/acl>; rel="acl"
```

This can be used when indexing repository content to determine what the access controls on the resource are.

ACLs for the Repository Root



When creating an ACL to protect the repository root, you **must** include a trailing slash in the Authorizations's `acl:accessTo` predicate, otherwise the Authorization will not match the request URI, and won't get applied.

Non-Working Version

```
<> a acl:Authorization;  
    acl:accessTo <https://localhost:8080/fcrepo/rest> .
```

Working Version

```
<> a acl:Authorization;  
    acl:accessTo <https://localhost:8080/fcrepo/rest/> .  
    # note this trailing slash -----^
```