# Web Access Control

Web Access Control (WebAC or WAC) is Fedora's system for authorizing requests for resources in the repository.

## Definitions

From the SOLID Web Access Control specification:

> *Web Access Control (WAC) is a decentralized cross-domain access control system. The main concepts should be familiar to developers, as they are similar to access control schemes used in many file systems. It's concerned with giving access to agents (users, groups and more) to perform various kinds of operations (read, write, append, etc) on resources. WAC has several key features:*
>
> 1. *The resources are identified by URLs, and can refer to any web documents or resources.*
> 2. *It is* declarative -- *access control policies live in regular web documents, which can be exported/backed easily, using the same mechanism as you would for backing up the rest of your data.*
> 3. *Users and groups are also identified by URLs (specifically, by WebIDs)*
> 4. *It is* cross-domain -- *all of its components, such as resources, agent WebIDs, and even the documents containing the access control policies, can potentially reside on separate domains. In other words, you can give access to a resource on one site to users and groups hosted on another site.*

WebAC enforces access control based on the Access Control List (ACL) RDF resource associated with the requested resource. In WebAC, an ACL consists of a set of Authorizations. Each Authorization is a single rule for access, such as "users alice and bob may write to resource foo", described with a set of RDF properties. Authorizations have the RDF type `http://www.w3.org/ns/auth/acl#Authorization` .

For the remainder of this document, the `http://www.w3.org/ns/auth/acl#` namespace will be abbreviated with the prefix `acl:`.

### Access Control Lists (ACLs)

An ACL is an RDF document (RDFSource) that contains WebAC statements that authorize access to repository resources. Each resource may have their own ACL, or implicitly be subject to the ACL of a parent container. The location of the acl for a given resource may be discovered via a `Link` header with relation `rel=acl`.

```
$ curl -I http://localhost:8080/fcrepo/rest/myContainer

Date: Thu, 23 Aug 2018 14:46:46 GMT
Expires: Thu, 01 Jan 1970 00:00:00 GMT
ETag: W/"919bed096330d23b2e85c01d487758aa6bbf2dcb"
Last-Modified: Thu, 16 Aug 2018 18:49:54 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"
Link: <http://localhost:8080/fcrepo/rest/myContainer/fcr:acl>; rel="acl"
Preference-Applied: return=representation
Vary: Prefer

...
```

If a resource does not have an individual ACL (and therefore relies on an implicit ACL from a parent), this link header will still be present, but will return a 404. This is because the location of ACLs is solely determined by the server, much like the automatically-created LDP-RS descriptions for binary resources. The key difference is that Fedora does not create ACLs automatically, only their location.

Therefore, to discover whether a resource has an individual ACL, a client would need to:

1. Perform a `HEAD` or `GET` against the resource,
2. Find the link header
3. Do a `GET` or `HEAD` against the ACL location, and see if returns 200 or 404.

To create an ACL for a resource that does not already have one, a client needs to discover the ACL location (via `HEAD` or `GET`), then `PUT` to that location.

### Authorizations

An ACL should contain one or more authorizations. Each authorization should have a hash URI resource as its subject, and an `rdf:type` of `http://www.w3.org/ns/auth/acl#Authorization`:

**Authorization**

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

<#auth1> a acl:Authorization .
```

The properties that may be used on an `acl:Authorization` are:

| Property | Meaning |
|---|---|
| `acl:accessTo` | The URI of the protected resource. |
| `acl:accessToClass` | An RDF class of protected resources. (*While the WebAC specification does not support acl:accessToClass, servers are required to support it according to the Fedora specification)* |
| `acl:agent` | The user *(in the W3C WebAC ontology, the user is named with a URI, but Fedora's implementation supports both URI- and string-based usernames)* |
| `acl:agentClass` | A class of agents, rather than a specific agent. Usage according to the WebAC specification is limited to `foaf:Agent` (meaning "everybody"), and `acl:AuthenticatedAgent` (meaning "any authenticated agent"). |
| `acl:agentGroup` | A group of users (defined as a `vcard:Group` resource listing its users with the `vcard:hasMember` property). |
| `acl:default` | Signifies that an authorization for a container may be inherited by children of that container, if they do not otherwise define their own ACLs. |
| `acl:mode` | The type of access (WebAC defines several modes: `acl:Read`, `acl:Write`, `acl:Append`, and `acl:Control`). |

For a more detailed explanation of Authorizations and their properties, see WebAC Authorizations.

## Agents

Agents are the users of Fedora.  These identify the principals (in a security sense) have made authenticated requests to the repository.  In ACL Authorizations used by Fedora, these may be represented as strings or as URIs.  The SOLID WebAC spec stipulates that agents are identified by URIs, and suggests (but does not have any normative language requiring) that these URIs are intended to be WebIDs.   The Fedora specification does not comment on the topic of identifying agents.  Nevertheless, for legacy purposes, the Fedora 5.x software allows strings or URIs to identify agents (e.g. `"bob"` or `<http://example.org/people/bob>`).  When using URIs, there is no expectation be Fedora that these URIs be resolvable, or have a representation.  *It is highly recommended that you use URIs*

The mapping of a logged-on principal to a string or URI depends on the selection and configuration of a Principal Provider, which may provide the identity of users as strings or URIs depending on its implementation.  Because agents are recommended to be represented as URIs, Fedora can be configured to automatically prefix any principals that are provided as strings with a baseURI.  This is achieved by setting the system property `fcrepo.auth.webac.userAgent.baseUri`.  For example:

**agent prefix**

```
fcrepo.auth.webac.userAgent.baseUri=http://example.org/agent/
```

Continuing with this example, if a user comes in as user "dra2", the user's identity will be converted to the URI http://example.org/agent/dra2 before applying ACLs.

# Examples of Authorizations

1. The user userA can Read document foo

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

<#auth1> a acl:Authorization ;
    acl:accessTo </fcrepo/rest/foo> ;
    acl:mode acl:Read;
    acl:agent "userA" .
```

2. Users in NewsEditor group can Write to any resource of type ex:News

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

<#auth2> a acl:Authorization ;
    acl:accessToClass ex:News ;
    acl:mode acl:Read, acl:Write;
    acl:agentClass </fcrepo/rest/agents/NewsEditors> .
```

**/agents/NewsEditors**

```
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .

<> a vcard:Group;
    vcard:hasMember "editor1", "editor2".
```

3. The user userB can Read document foo (This involves setting a system property for the servlet container, e.g. `-Dfcrepo.auth.webac.userAgent.baseUri=http://example.org/agents/`)

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

<#auth3> a acl:Authorization ;
    acl:accessTo </fcrepo/rest/foo> ;
    acl:mode acl:Read;
    acl:agent <http://example.org/agents/userB> .
```

## Protecting Resources

Any resource in the repository may have its own ACL. The location of that (potential) ACL is given in a `Link` HTTP header with `rel="acl"`. If a resource itself does not specify its own ACL, its parent containers are inspected, and the first specified ACL found is used as the ACL for the requested resource. If no ACLs are found, a filesystem-based ACL will be checked, the default policy of which is to deny access to the requested resource.

The standard location for a resource's ACL is the `fcr:acl` child of that resource, but clients should not rely on this behavior and always "follow their nose" by checking the `Link` header.

## How-To Guides

- Quick Start with WebAC
- How to Use WebAC Groups
- WebAC Example Scenarios

## More Detailed Documentation

- SOLID WebAC Specification
- Determining the Effective Authorization Using WebAC
- W3C's WebAC Ontology