WebAC Authorizations

In WebAC, an Authorization is a rule that describes who can access what, and how they can interact with the resource they can access. An Authorization is written as a series of RDF triples, with the Authorization resource as the subject for each of these triples.

Prefixes used in this document:

```
@prefix acl:   <http://www.w3.org/ns/auth/acl#> .
@prefix ex:    <http://example.org/ns#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix ldp:   <http://www.w3.org/ns/ldp#> .
@prefix pcdm:  <http://pcdm.org/models#> .
@prefix webac: <http://fedora.info/definitions/v4/webac#> .
```

"Who?" - Users and Groups

The first part of the authorization describes who can access a resource. There are two ways to do this. The first is simply by naming particular users using the `acl:agent` property, as either strings or URIs. When using URIs, it is necessary to translate the string-based username that is used for container-based authentication (i.e. Tomcat/Jetty authentication) into a URI. This is done by prefixing the string with a URI that is set in the container configuration: `-Dfcrepo.auth.webac.userAgent.baseUri=http://example.org/user/` and/or `-Dfcrepo.auth.webac.groupAgent.baseUri=http://example.org/group/`

```
# as strings
<> acl:agent "obiwan", "yoda" .

# as URI references
<> acl:agent ex:obiwan, ex:yoda .
```

*(**Fedora Implementation Note:** This is a slight departure from the W3C's description of WebAC, where the object of the `acl:agent` property must be a URI. We chose to implement it in such a way that the WebAC authorization module supports both String Literals and URIs in order to ease the integration with existing authentication or single-sign-on systems that identify users with string usernames.)*

However, listing individual users this way can get unwieldy, so you can also use the `acl:agentClass` property to specify a group of users:

```
<> acl:agentClass </groups/jedi> .
```

The object of the `acl:agentClass` property must be a URI to a group resource (of type `foaf:Group`) containing a list of its members:

```
# contents of </groups/jedi>, using strings to identify members:
<> a foaf:Group;
    foaf:member "obiwan";
    foaf:member "yoda";
    foaf:member "luke" .

# contents of </groups/jedi>, using URIs to identify members:
<> a foaf:Group;
    foaf:member ex:obiwan;
    foaf:member ex:yoda;
    foaf:member ex:luke .
```

*(**Fedora Implementation Note:** As currently implemented, the group resource must also be stored in Fedora; there is no support for referencing external URIs with the `acl:agentClass` property.)*

"What?" - Resources and Resource Types

The next part of the authorization describes what resource can be accessed. As with the previous section on agents, there are also two ways to describe the resource. The first is to provide the URI to the resource using the `acl:accessTo` property:

```
<> acl:accessTo </collections/rebels/plans> .
```

If you use `acl:accessTo` to protect a container, that authorization rule by default will also apply to any of that container's children, unless that child has its own `acl:accessControl` property, as described below.

The second is to use the `acl:accessToClass` property to state that the authorization rule applies to any resource with the named RDF type:

```
# this will apply to any pcdm:Container resources
<> acl:accessToClass pcdm:Container .
```

Note that adding `acl:accessTo` or `acl:accessToClass` to an authorization is only one half of what is required to protect a resource with a WebAC ACL. The other half is to specify on the resource itself that it is protected by the ACL that contains that authorization:

```
# </acls/rebels>
<> a webac:Acl;
    ldp:contains <commanders>.

# </acls/rebels/commanders>
<> a acl:Authorization;
    acl:agentClass </groups/rebel-commanders>;
    acl:accessTo </collections/rebels/plans>;
    # modes will be discussed in the next section
    acl:mode acl:Read, acl:Write.

# partial contents of </collections/rebels/plans>:
<> acl:accessControl </acls/rebels> .
```

"How?" - Modes of Interaction

Finally, an authorization states how the users or groups can interact with the resource or type of resource. This is known as the mode of access, and is specified using the `acl:mode` property. There are two predefined modes from the W3C's description of WebAC that Fedora understands: `acl:Read` and a `cl:Write`. There are two additional access modes defined by the W3C document: `acl:Control` and `acl:Append`; however, they are not implemented in the fedora WebAC module.

Bringing it All Together

Here is a complete example of a WebAC ACL and its authorizations:

```
# </acls/rebels>
<> a webac:Acl;
    ldp:contains <commanders-plans>
    ldp:contains <pilots-plans>;
    ldp:contains <pilots-flight-plans>.

# </acls/rebels/commanders-plans>
# commanders...
<> a acl:Authorization;
    # ...listed in this group...
    acl:agentClass </groups/rebel-commanders>;
    # ...have read-write access to...
    acl:mode acl:Read, acl:Write;
    # ...the plans
    acl:accessTo </collections/rebels/plans>.

# </acls/rebels/pilots-plans>
# but pilots...
<> a acl:Authorization;
    # ...listed in this group...
    acl:agentClass </groups/rebel-pilots>;
    # ...have read-only access to...
    acl:mode acl:Read;
    # ...the plans
    acl:accessTo </collections/rebels/plans>.

# </acls/rebels/pilots-flight-plans>
# however, pilots...
<> a acl:Authorization;
    # ...listed in this group...
    acl:agentClass </groups/rebel-pilots>;
    # ...do have read-write access to...
    acl:mode acl:Read, acl:Write;
    # ...their flight plan documents
    acl:accessToClass ex:FlightPlan.

# </collections/rebels/plans>
# this resource is protected by the ACL at this URI
<> acl:accessControl </acls/rebels> .

# </collections/rebels/flights>
# this collection also specifies an ACL so all of its child resources will be
# covered by an ACL
<> acl:accessControl </acls/rebels>;
   ldp:contains <trench-run>.

# </collections/rebels/flights/trench-run>
# users in the group rebel-pilots will have read-write access to this resource
<> a ex:FlightPlan.
```