


# FeSL Authorization

## FeSL Authorization

FeSL Authorization is based on XACML version 2. XACML policies are stored in the Fedora repository as FESLPOLICY datastreams on Fedora objects. These datastreams can be either inline XML ("X") or managed content ("M").

A set of bootstrap system policy objects are created when Fedora first starts, from the policies in the `$FEDORA_HOME/pdp/policies` directory. These objects have a `fedora-policy` PID namespace.

### Modifying Bootstrap Policies

 Bootstrap policy objects are not updated if you modify the policies in this directory. If you need to modify a bootstrap policy the FESLPOLICY datastream of the `fedora-policy` object must be modified.

- [FeSL Authorization](#)
- [Configuration](#)
  - [Policy evaluation results caching.](#)
  - [Configuration files](#)
- [FeSL XACML Policies](#)
  - [Specifying resources in policies](#)
  - [The XACML hierarchical resource profile](#)
  - [XACML action attributes](#)
  - [Policies based on relationships](#)
    - [Resource attributes based on object and datastream properties.](#)
    - [Resource attributes based on more complex Resource Index queries.](#)
  - [Relationships and dependency on the Resource Index](#)
- [Disabling FeSL Authorization](#)

## Configuration


### Policy evaluation results caching.

Results of previous policy evaluations are cached. This is used to improve performance so that multiple requests to the same resource do not require multiple evaluations of the same set of XACML policies.

If a policy is updated, the cache is not cleared. The items in the cache are cleared when they expire, based on the values configured in the `response-cache` element in `$FEDORA_HOME/server/config/config-melcoe-pep.xml`. So a cached evaluation result from a previous request may be returned rather than evaluation being based on the updated policy.

Evaluation response caching can be disabled by setting the system property `fedora.fesl.pep_nocache=true`. This can be done when starting Tomcat by first adding `-Dfedora.fesl.pep_nocache=true` to the `CATALINA_OPTS` environment variable.

Environment variable `PEP_NOCACHE` is deprecated

 In previous releases of Fedora up to and including version 3.5, the response cache could be disabled by setting the environment variable `PEP_NOCACHE=true`. Although this configuration is still recognised you are recommended to use the `fedora.fesl.pep_nocache` system property instead, as this configuration option may be removed in a future release.

### Configuration files

Configuration files are located in `FEDORA_HOME/pdp/conf`

- `config-pdp.xml`  
This is the dynamic configuration file for the Sun XACML PDP Evaluation Engine. This file is used to register PolicyFinder modules, AttributeFinder modules and ResourceFinder modules
- `config-policy-storage.xml`  
This file configures the back-end policy index. Refer to the installation instructions for how to configure various policy index implementations. In addition this configuration file is used to specify which policy combination algorithm to use when multiple policies are retrieved.
- `config-pdm-fedora.xml`  
This file contains settings for how Fedora policy objects are created for the bootstrap policies loaded when the server is accessed for the first time.
- `config-attribute-finder.xml`  
-This config file is for the Fedora RISearch Attribute finder. When policies need additional information from Fedora that was not provided in the XACML Request, they can be retrieved by custom AttributeFinders.

## FeSL XACML Policies

FeSL XACML policies are implemented as FESLPOLICY datastreams on Fedora objects. You may choose to have stand-alone policy objects, or may add FESLPOLICY datastreams to existing objects. FESLPOLICY datastreams can be inline XML (X) or managed (M) (external datastreams will also be recognised, but modifications to these datastreams cannot be detected, and therefore the policy index will not reflect changes in the datastreams, therefore use of external datastreams are not recommended).

## Specifying resources in policies

Unlike the POLICY datastream used in the legacy XACML implementation, there is no implicit relationship between the policy specified in the FESLPOLICY datastream and the object in which the datastream resides. The FESLPOLICY datastream does not by default apply to the containing object - you must specify the resource(s) to which the policy applies in the XACML.

## The XACML hierarchical resource profile

FeSL supports the path-based implementation of the XACML 2.0 [Hierarchical Resource Profile](#).

The path used for the hierarchical resource profile resource IDs is the full path to the resource using collection/member relationships specified in RELS-EXT. The relationships used in building the hierarchical path are configured in \$FEDORA\_HOME/server/config/config-melcoe-pep.xml. The default relationships specified are those used in the Fedora relationships ontology.

The XACML policy should specify the ResourceAttributeDesignator as urn:oasis:names:tc:xacml:1.0:resource:resource-id and should use a ResourceMatch MatchID of urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match.

You may specify policies applying to collections of objects using the following syntax for resource attribute values:

- /pid:collection/. \* - applies to all children and descendents of collection object pid:collection
- /pid:collection1/pid:collection2/. \* - applies to all children and descendents of collection object pid:collection2 (but only when pid:collection2 is itself a member of pid:collection1)

The **full path** of the collection hierarchy must be specified. So where you have collections that are themselves children of other collections as in the second example above you must supply the full path to the child collection. (The XACML specification allows for full regular expression syntax in specifying the hierarchical path, however due to difficulty of indexing the full regular expression syntax, only trailing wildcards (.\*) are supported.)

To apply the policy to the collection object itself as well as its children you will need to specify the collection object separately, ie

- a resource AttributeValue of /pid:collection
- a ResourceMatch MatchID of urn:oasis:names:tc:xacml:1.0:function:anyURI-equal

Example:

```
<Resources>
  <Resource>
    <!-- to view everything under the resource collection -->
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">/demo:
demoObjectCollection/. *</AttributeValue>
      <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:
resource-id" DataType="http://www.w3.org/2001/XMLSchema#anyURI" />
    </ResourceMatch>
  </Resource>
  <Resource>
    <!-- to view the resource collection itself-->
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">/demo:
demoObjectCollection</AttributeValue>
      <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:
resource-id" DataType="http://www.w3.org/2001/XMLSchema#anyURI" />
    </ResourceMatch>
  </Resource>
</Resources>
```

Multiple hierarchies are not supported

FeSL does not support hierarchical policy evaluation where objects are members of more than one collection; ie where there are multiple paths to the same resource.

Policies can be applied to more than one collection in a hierarchy of collections - for instance if an object is a member of pid:collection1 which is itself a member of pid:collection2, policies can be specified for both of these collections. Policies are evaluated and combined according to the policy combining algorithm specified in \$FEDORA\_HOME/pdp/conf/config-policy-storage.xml. The default combining algorithm specifies that the policies at lower levels override those at highest levels, with "deny" overriding "permit".

## XACML action attributes

Action attributes specify which actions the policy applies to. In the legacy XACML implementation, Fedora supports specifying attribute values for individual API calls, and generic values to specify actions at the level of API-M and API-A.

FeSL supports a simplified set of action attribute values of create, read, update and delete, in addition to the above. The relationship of these values to the underlying API action attribute are specified in `$FEDORA_HOME/server/config/config-melcoe-pep-mapping.xml`.

## Policies based on relationships

FeSL supports the ability to define policies based on relationships expressed in the RELS-EXT and RELS-INT datastreams. The target of a relationship from an object or a datastream can be defined as a XACML resource attribute.

As well as defining XACML resource attributes based on simple object and datastream relationships, more complex specifications can be defined using Resource Index queries.

These resource attributes are defined in `$FEDORA_HOME/pdp/conf/config-attribute-finder.xml`

## Resource attributes based on object and datastream properties.

Example:

```
<attribute designator="resource" name="http://www.example.org/fedora/xacml/attributes/resource#rel-subject">
  <config name="relationship" value="http://purl.org/dc/elements/1.1/subject"/>
  <config name="target" value="object"/>
</attribute>
```

In the example above, the `dc:subject` relationship is made available as a XACML resource attribute.

- XACML resource attribute ID: this is defined by the "name" attribute, in this case <http://www.example.org/fedora/xacml/attributes/resource#rel-subject>
- Fedora RI relationship: this is defined by the "relationship" config element. In this case the relationship used is the `dc:subject` relationship, <http://purl.org/dc/elements/1.1/subject>
- Subject of relationship query: The "target" configuration element specifies the subject of the RI query. In this case the value "object" is specified which means the RI query is based on the URI of the Digital Object containing the resource being accessed. This means that the `dc:subject` attribute of the digital object will be retrieved when both the object itself is accessed and when an object datastream is accessed. If this value is specified as "resource" (or is omitted) the URI of the resource (eg a datastream) being accessed will be used instead. Use "resource" where you want to base an attribute on a datastream property defined in RELS-INT.

If the "relationship" config element is omitted, the same value will be used for the Fedora RI relationship and the XACML resource attribute ID (for instance <http://purl.org/dc/elements/1.1/subject> could be defined as both the XACML resource attribute ID to be used in policies and the RI relationship to be used in retrieving attribute values.

## Resource attributes based on more complex Resource Index queries.

XACML resource attributes can be defined by specifying RI queries using any language that the Resource Index supports. Mulgara supports:

- itql
- sparql
- spo

Example:

```
<attribute designator="resource" name="http://www.example.org/fedora/xacml/attributes/resource#collection-owner">
  <config name="queryLang" value="itql"/>
  <config name="object" value="##object##"/>
  <config name="value" value="owner"/>
  <config name="query" value="select $collection $owner from <#ri> where
  <##object##> <#info:fedora/fedora-system:def/relations-external#isMemberOfCollection> $collection
  and $collection <#info:fedora/fedora-system:def/model#ownerId> $owner"/>
</attribute>
```

- XACML Resource Attribute ID: specified by the "name" attribute.
- RI Query Language: specified by the "queryLang" config element
- config elements "resource" and "object" specify the text pattern in the query to substitute with the URI of the resource being accessed. Either or both of these config elements can be specified.
  - The value for "resource" will be substituted with the URI of the resource being accessed (so if a datastream is being accessed, the URI of the datastream)

- the value for "object" will be substituted with the URI of the containing digital object (so if a datastream is being accessed, the URI of the object containing the datastream).
- The "value" config element specifies the output variable of the query which supplies the XACML resource attribute values.
- The "query" config element specifies the RI query to use. Note that the query must be xml-escaped; ie < must be substituted with < and > must be substituted with >.

In the above example the unescaped itql query is:

```
select $collection $owner
from <#ri>
where
<##object##> <info:fedora/fedora-system:def/relations-external#isMemberOfCollection> $collection
and
$collection <info:fedora/fedora-system:def/model#ownerId> $owner
```

When the datastream "DC" is accessed in the object "demo:1", a policy using the XACML resource attribute ID <http://www.example.org/fedora/xacml/attributes/resource#collection-owner> will be evaluated as follows:

- the text ##object## will be substituted with the URI of the digital object containing the resource being accessed, in this case info:fedora/demo:1
- the query follows the collection membership relationship from the object to its parent, returning this as the \$collection output variable
- the ownerId property of the parent is returned as the \$owner output variable
- the value(s) of the \$owner output variable are returned to the policy evaluation engine for comparison with the values specified in the XACML policy.

Thus in this example it is possible to control access to an object and its datastreams based on the ownerId of the collection to which the object belongs.

Example policy fragment:

```
<Rule Effect="Permit" RuleId="1" >
  <Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
      <ResourceAttributeDesignator AttributeId="http://www.example.org/fedora/xacml/attributes/resource#collection-owner" DataType="http://www.w3.org/2001/XMLSchema#string"></ResourceAttributeDesignator>
      <SubjectAttributeDesignator AttributeId="urn:fedora:names:fedora:2.1:subject:role" DataType="http://www.w3.org/2001/XMLSchema#string"></SubjectAttributeDesignator>
    </Apply>
  </Condition>
</Rule>
```

In the above example XACML Rule, access to an object is permitted if the user accessing the resource has a role equal to the ownerId property of any collection to which the object belongs. Or, a user has read access to collection members where the user's role is the "owner" of the collection.

The FunctionID "urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of" does a "bag" comparison - the function evaluates to true if there is at least one match in the two bags being compared. In this case the first bag is all of the ownerId attributes of all collections to which the resource belongs, and the second bag is the all of the role attributes of the subject.

## Relationships and dependency on the Resource Index

By default, FeSL queries the Resource Index for relationship information. It does this when querying for parent/child relationships for policies based on the XACML hierarchical resource profile, and for policies based on resource attributes defined using SPO and more complex RI queries.

If you do not have the Resource Index enabled, you may instead configure FeSL to use RELS-EXT and RELS-INT datastreams directly to derive relationship information.

This is configured in `$FEDORA_HOME/server/config/config-melcoe-pep.xml`.

Change the class attribute of the relationship-resolver element to `org.fcrepo.server.security.xacml.util.RELSRelationshipResolver` to use the relationships datastreams directly.

Note that this introduces some restrictions on FeSL's features

- Collection-based policies based on the XACML hierarchical resource profile will require that relationships are specified from the child object to the parent (eg isMember relationships in the child object)
- Only simple datastream and object properties can be exposed as XACML resource attributes; the properties must be defined as relationships in the containing object

## Disabling FeSL Authorization

If you encounter problems, such as creating a set of policies which lock out administrative access to the repository and thus prevent further changes to policies, you can disable FeSL AuthZ completely.

1. Locate the file `$FEDORA_HOME/server/config/spring/web/security.xml`.
2. Make a backup copy of this file
3. For each of the `<security:filter-chain>` elements, remove the value `PEPFilter` (and the preceeding comma) from the `filters` attribute.
4. When you have finished correcting any problems, re-instate the backup copy of this file.