Implementing XACML policies for ETD and NJVid

Introduction

This document will set forth a recommendation for XACML to control the access for video data stream based on users' role for NJVid; for XACML to embargo and restrict the accessing of the supplemental files or Dissertations for ETD. It describes the content model structure to include a POLICY datastream. It also describes the requirements for WMS to implement XACML in the future release for ETD and NJVid. It finally gives examples of XACML POLICY datastream for ETD and NJVid object.

Fedora XACML Basic

Release 5.0 shall have Fedora 3.0 up in production. Since Fedora 2.1b introduces the security architecture feature - a XACML-based policy enforcement module, we are going to use XACML for ETD and NJVid after Fedora 3.0 in place. XACML is an XML-based markup language to program access control policies. The policy language can be used to control access to Fedora web services, Fedora digital objects, datastreams, disseminations, and more.

The Authorization module is built upon the Sun XACML engine. Each XACML policy defines:

(1) A "target" describes what the policy applies to (by referring to attributes of users, operations, objects, datastreams, dates, and more)

(2) One or more "rules" to permit or deny access.

Policy [Rule Combining Algorithm]

(1) Target
- Subject Attributes
- Resource Attributes
- Action Attributes
- Environment Attributes

(2) Rule [Effect]
- Subject Attributes
- Resource Attributes
- Action Attributes
- Environment Attributes
- Conditions

There is a Fedora-specific policy vocabulary for referring Fedora operations and Fedora-specific entities within XACML policies.

Fedora supports both repository-wide policies (that specify broad access controls that apply to the entire repository) and object-specific policies (that specify rules for a single object, and can even be stored within the object in a special datastream).

There are three ways to store the XACML policy.

1. Repository-wide policies are stored in Fedora reserved location
2. Object-specific policies are stored in Fedora reserved location
3. Object-specific policies are stored inside digital objects

Revised object architecture

In our scenario, an object-specific policy shall be stored inside a digital object within the special reserved datastream whose ID is "POLICY".

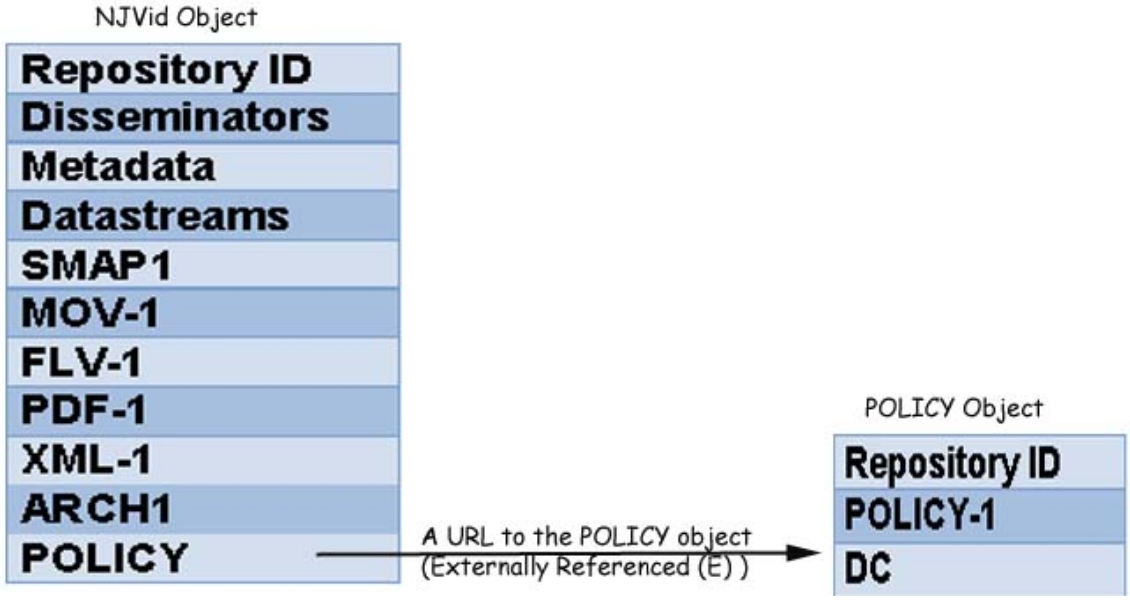
There are several benefits to store object-specific policies in the POLICY datastream.

Firstly, the policy is only evaluated for this Fedora API requests that pertain to the digital object in which the policy "resides." Policies that are stored in the POLICY datastream are not loaded into memory upon startup of the Fedora server (as are the repository-wide policies). Therefore, the POLICY datastream approach may be a way to enhance performance in cases where a large number of object-specific policies exist.

Secondly, the benefit is that the POLICY datastream storage strategy may provide for easier distribution of policy management responsibilities. For example, authors or owners of particular digital objects can be granted the rights to view and modify the POLICY datastream of their objects, without having to obtain repository administrator privileges to modify policies in a configured policy storage location external to a repository.

Thirdly, another benefit of putting object-specific policies in the POLICY datastream in by-reference is that multiple digital objects can point to the same policy. You can store the policy external to the digital objects to which it pertains, and store the URL of the external policy location in the E or R type datastream. Fedora resolves such URLs at runtime, so when the policy enforcement module needs to evaluate the XACML policy, Fedora will automatically retrieve the policy file from its external location and pass it on the policy enforcement module for evaluation. Policies as Externally Referenced (E) or Redirected (R) Datastreams become portable with their parent digital objects (e.g., the policies goes with an object, as datastream content, when the object is exported from the repository).

The POLICY datastream could point to a local file or a policy object. See the resulting architecture for NJVid and the POLICY object as follows:



Note: This POLICY object shall only have a DC datastream and shall not be indexed.

If POLICY datastream is stored in line, Fedora admin shall be able to view and edit the POLICY datastream in development environment server.

Requirements for WMS in the feature release

XACML for both NJVid and ETDs shall be added into the object after the object has been ingested into Fedora. For NJVid, the XACML will be in a separate object and pointed to by the POLICY datastream in the video object. The URL is marked as "E". For ETDs, the XACML will be xml inline as part of the ETD object. The procedure as follows: We create XACML manually; then we add POLICY datastream through the DLR EDIT after Fedora 3.0 in place.

During the R5.1, WMS shall implement a POLICY pick list to let the users to pick up the POLICY datastreams according to users' needs from a list of external POLICY datastreams which have been created and stored in Fedora system already.

Possible addition for WMS is to develop an access control admin tool. The access control admin tool should let the admin user generate the POLICY datastream on the fly based on the selection of target, users or groups, permissions (get, modify, delete), and environment etc for particular object.

Prototype XACML policies for ETD

Assuming there are two resources*:* datastreams namely:

1. Supplemental files
2. Dissertations

For ETD, we shall use XACML to embargo and restrict the accessing of the supplemental files or Dissertations. Embargo XACML POLICY would be a standard template with only the "embargo expiration date" need to be modified. The POLICY shall be inline POLICY and manually created in Fedora object for release 5.0.

In ETD scenario, let's assume student would like to embargo the dissertation in PDF format for a year after the graduation date. The embargo date is in UTC format. Example here: one year after 2008 May 11th will be 2009 May 11th. We compare the embargo date with the current date in the code below. If the Condition evaluates to true, then the Rule's Effect.

Once a Policy has been found and verified to apply to a request, its Rules are evaluated. A policy can have any number of Rules which contain the core logic of an XACML policy. The heart of most Rules is a Condition, which is a boolean function. If the Condition evaluates to true, then the Rule's Effect (a value of Permit or Deny that is associated with successful evaluation of the Rule) is returned. Below is the example of the ETD embargo XACML.

```
<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than">
<EnvironmentAttributeDesignator
AttributeId="urn:fedora:names:fedora:2.1:environment:currentDateTime"
DataType="http://www.w3.org/2001/XMLSchema#dateTime"/>
```

Comment: ETD embargo time in UTC format, **2009-05-11T15:11:06.502Z.** Example is one year after graduation date 2008 May 11th.

```
<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#dateTime">2009-05-
11T15:11:06.502Z</AttributeValue>
</Apply>
 </Condition>
```

Comment: ETD resource – dissertation, in our case it might be a datastream ID such as PDF-1

```
 <Resource>
 <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
```

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">PDF-1</AttributeValue>

<ResourceAttributeDesignator

AttributeId="urn:fedora:names:fedora:2.1:resource:datastream:id"

DataType="http://www.w3.org/2001/XMLSchema#string">

</ResourceAttributeDesignator>

</ResourceMatch>

</Resource>

Prototype XACML policies for NJVid

Assuming there are four different roles for NJVID namely:

1. Student
2. Member of Organization
3. Faculty
4. Admin

There are two resources: datastreams namely:

3. QuickTime video file
4. Flash Movie

For each role and resource we explicitly mention the access control decision in form of a table as shown below:

Role	Video File	Action	Purpose
Student	Deny	getDatastreamDissemination	Restricted Access
Member	Deny	getDataStreamDissemination	Restricted Access
Faculty	Deny	getDataStreamDissemination	Restricted Access
Admin	Permit	getDataStreamDissemination	Moderation

In NJVid scenario, let's assume student and faculty has authorized to access the NJVid. But student only allow viewing and streaming the public video and faculty can view and stream both the public and copyright video. We shall have a XACML POLICY to deny the student to access the copyright video.

As XACML always expresses policies in terms of **Subject** that can do an **Action** on a **Resource**, we need information about these three items.

- **Subjects** - Information about a subject i.e. the user who makes the request is wrapped within a <Subjects> tag. The <SubjectMatch> tag is used to match information of the subject with the requirements of the policy by applying different matching and comparison functions that XACML specification provides. In this case we match the role of the subject to student. This ascertains that this policy is targeted to all users having a role student.

```
<Subjects>
   <Subject>
    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">student</AttributeValue>
      <SubjectAttributeDesignator AttributeId="fedoraRole" MustBePresent="false"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
   </Subject>
 </Subjects>
```

- **Actions** - Actions like getDatastream, getDissemination getDatastreamDissemination to be performed is wrapped within the <Actions>tag. The action is matched to view by using the <ActionMatch> tag as done in the policy.

```
<Action>
      <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">urn:fedora:names:fedora:2.1:actio
n:id-getDatastreamDissemination</AttributeValue>
      <ActionAttributeDesignator DataType=http://www.w3.org/2001/XMLSchema#string
AttributeId="urn:fedora:names:fedora:2.1:action:id"/>
      </ActionMatch>
</Action>
```

- **Resources -** As mentioned the policy is applicable to the request of NJVid resources namely video files. If a request for any other resource apart from these is made; then the policy will not be applicable. The resource information and matching is done within the <Resources> tag by using the <ResourceMatch> tag.

```
<Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">MOV-
                1</AttributeValue>
                <ResourceAttributeDesignator
                AttributeId="urn:fedora:names:fedora:2.1:resource:datastream:id"
                DataType="http://www.w3.org/2001/XMLSchema#string">
                </ResourceAttributeDesignator>
        </ResourceMatch>
</Resource>
```

- **Attribute Matching Logic** - The match logic used in all the cases above is: MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal". It compares two string values and returns a boolean value stating wheter the strings match or not. The full list of all available functions can be seen in the XACML Core Documentation.

Issues and Recommendations about XACML policies

Recommendations:

1. XACML shall be manually created for both NJVid and ETDs. Then shall be added into the object after the object has been ingested into Fedora for release 5.0. For NJVid, the xacml will be in a separate object and pointed to by the POLICY datastream in the video object. The URL is marked as "E". For ETDs, the XACML will be xml inline as part of the ETD object. The URL is marked as 'X" and the format will be FOXML.
2. We are not going to extend XACML POLICY to sun's XACML for release 5.0. Fedora has specific policy vocabulary for referring Fedora operations and Fedora-specific entities within XACML policies. If extend to sun's XACML, it will have more flexibility. Depends on the ETD and NJVid's requirement, if Fedora XACML vocabulary is enough to deal with all the actions ETD and NJVid needs, then we can stay with Fedora's XACML.
3. DLR EDIT shall make changes to make the POLICY datastream visible after Fedora 3.0 in place. Administrator shall be able to edit and view the POLICY.

Issues:

1. If Shibboleth is the choice, how can we pass the Shibboleth authentication information to Fedora?

This implies that we need to find a way to authenticate the GUI user to Fedora. Fedora needs to have an understanding of users and their attributes (like roles, email, name, etc.) in order for policy evaluation. Fedora would subsequently inform what permissions the user has. Attached is the Fedora user's database in XML format.

```
<?xml version="1.0" encoding="UTF-8"?>
<users>
    <user name="fedoraAdmin" password="fedoraAdmin">
        <attribute name="fedoraRole">
            <value>administrator</value>
        </attribute>
    </user>
    <user name="Student" password="changeme" >
        <attribute name="fedoraRole">
            <value>RU-student</value>
        </attribute>
    </user>
</users>
```

jg – 5/29/2008