Hierarchical Resources: Non-XML Resource Use Case

Working Draft 01, 17 June 2004

Document identifier:

xacml-profile-hierarchical-resources-nonXML-1.0-draft01

Location:

http://www.oasis-open.org/committees/documents.php?wg_abbrev=xacml

Editor:

Anne Anderson, Sun Microsystems (anne.anderson@sun.com)

Contributors:

Frank Siebenlist, Argonne National Laboratory
Daniel Engovatov, BEA
Hal Lockhart, BEA
Tim Moses, Entrust
Simon Godik, GlueCode Software
Bill Parducci, GlueCode Software
Michiharu Kudo, IBM
Michael McIntosh, IBM
Anthony Nadalin, IBM
Steve Anderson, Open Network
Seth Proctor, Sun Microsystems
Polar Humenn, Syracuse University

Abstract:

This document provides a use case using the XACML Profile for Hierarchical Resources with a hierarchical resource that is not an XML document.

Status:

This version of the document is a working draft of the Committee. As such, it is expected to change before becoming part of any standard. Committee members should send comments on this specification to the xacml@lists.oasis-open.org list. Others should use the comment form at http://www.oasis-open.org/committees/comments/form.php?wg_abbrev=xacml

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the XXXX TC web page (http://www.oasis-open.org/committees/xacml/ipr.php).

Table of Contents

1 Introduction

This document provides a use case and solutions for using XACML to protect a hierarchical resource that is not an XML document. The solutions make use of functionality described in the *XACML Profile for Hierarchical Resources* [HIER].

This document is non-normative. It provides guidance and suggestions, but does not set limits on valid uses of XACML.

1.1 Terminology

The policy examples in this document assume the following XML Internal Entities have been defined.

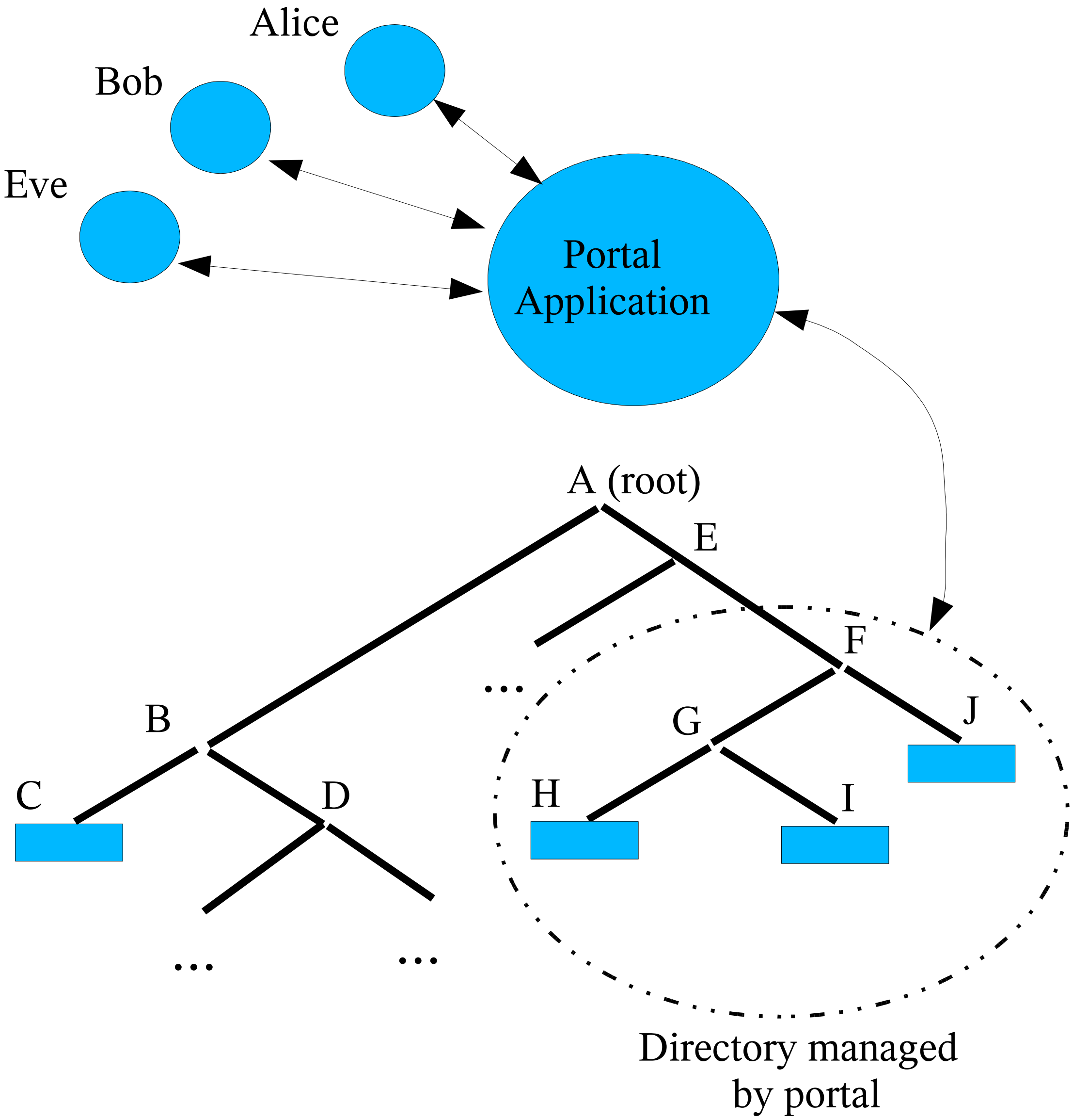
```
^lt;!ENTITY action-id "urn:oasis:names:tc:xacml:1.0:action:action-id">
^lt;!ENTITY and "urn:oasis:names:tc:xacml:1.0:function:and">
^lt;!ENTITY anyURI-match
              "urn:oasis:names:tc:xacml:2.0:function:anyURI-match”>
^lt;!ENTITY not "urn:oasis:names:tc:xacml:1.0:function:not">
^lt;!ENTITY or "urn:oasis:names:tc:xacml:1.0:function:or">
^lt;!ENTITY permit-overrides "urn:oasis:names:tc:xacml:1.0:rule-
              combining-algorithm:permit-overrides">
^lt;!ENTITY resource-ancestor
              "urn:oasis:names:tc:xacml:2.0:resource:resource-ancestor">
^lt;!ENTITY resource-id
              "urn:oasis:names:tc:xacml:1.0:resource:resource-id">
^lt;!ENTITY resource-parent
              "urn:oasis:names:tc:xacml:1.0:resource:resource-parent">
^lt;!ENTITY string "http://www.w3.org/2001/XMLSchema#string">
^lt;!ENTITY string-is-in
              "urn:oasis:names:tc:xacml:1.0:function:string-is-in">
^lt;!ENTITY string-one-and-only
            "urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
^lt;!ENTITY subject-id
              "urn:oasis:names:tc:xacml:1.0:subject::subject-id">
```

For example, “`&string`” is equivalent to “`http://www.w3.org/2001/XMLSchema#string`”.

2 Use Case: A Portal-Managed Directory

In this use case, a portal application serves as a front-end for client access to a directory in a Unix File System (UFS). The portal application is responsible for accessing objects in its directory on behalf of clients, but only if the access conforms to the enterprise policy. The portal application itself has full operating system permissions on objects in its directory, and has “execute” rights on the path from the root directory to its own directory. No other identities have rights in its directory, and the portal application has no rights in directories other than its own.

An illustration of the use case is shown in Drawing 1 below.



Drawing 1 Portal-Managed Directory

3 Problem Statement

We assume that the portal application has a Policy Enforcement Point (PEP) that traps each client attempt to access an object in the managed directory. The PEP sends a description of the access attempt to a Policy Decision Point (PDP), which compares the access attempt to its policies and returns an access decision of "Permit" or "Deny". The PEP enforces the access decision, and will perform the requested access for the client only if the access decision was "Permit".

The enterprise Policy Administration Point (PAP) will create the policies that will be used by the portal application's PDP in making its policy decisions. The problem is how to express these policies.

As a very simple example, assume the PAP wants to enforce the following rules:

- Anyone can "read" directory F. That is, anyone can view the identities of the immediate child objects in directory F. In addition,
- Alice can "read" any file or sub-directory under directory F except for file I
- Bob can "read" only files H and I.

This policy could be expressed as a decision table as follows:

Object	**Read Permission**
Directory F	anyone
Directory G	Alice
File H	Alice, Bob
File I	Bob
File J	Alice

The next sections will describe various ways of expressing this policy in XACML.

4 XACML Solutions

XACML provides several ways to solve the problem of expressing policies for the portal-managed directory. Some of these solutions are described below.

Note that in this use case, access control by XACML policies is limited by the privileges enforced by the operating system. For example, if the operating system does not allow the portal application to read a given file, then the portal application will be unable to read the file on behalf of a given client even if there is an XACML policy that would return “Permit” for that access. As another example, if the operating system grants a client access to read a file, then the portal application will be unable to prevent the client from reading the file directly (that is, not through the portal application).

4.1 Solution 1: Policy predicates for each directory object

The simplest solution is to include predicates in the XACML policy for each object in the managed directory. For each of the objects in the managed directory – directories F, G, and files H, **I**, and J – the policy will explicitly describe the conditions under which access is allows.

An example of a policy implementing the decision table shown in Section 3 follows.

```
<Policy PolicyId=”PortalPolicy” RuleCombiningAlgId=”&permit-overrides;”>
  <Target>
    <Actions>
      <Action>
        <ActionMatch MatchId=”&string-is-in;”>
          <AttributeValue DataType=”&string;”>read</AttributeValue>
          <ActionAttributeDesignator AttributeId=”&action-id;”
DataType=”&string;”/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Rule RuleId=”Directory:F” Effect=”Permit”>
    <Condition FunctionId=”&string-is-in;”>
      <AttributeValue DataType=”&string;”>F</AttributeValue>
      <ResourceAttributeDesignator AttributeId=”&resource-id;”
DataType=”&string;”/>
    </Condition>
  </Rule>
  <Rule RuleId=”Directory:G” Effect=”Permit”>
    <Condition FunctionId=”&and;”>
      <Apply FunctionId=”&string-is-in;”>
        <AttributeValue DataType=”&string;”>Alice</AttributeValue>
        <SubjectAttributeDesignator AttributeId=”&subject-id;”
DataType=”&string;”/>
      </Apply>
      <Apply FunctionId=”&string-is-in;”>
        <AttributeValue DataType=”&string;”>G</AttributeValue>
        <ResourceAttributeDesignator AttributeId=”&resource-id;”
DataType=”&string;”/>
      </Apply>
    </Condition>
  </Rule>
  <Rule RuleId=”File:H” Effect=”Permit”>
    <Condition FunctionId=”&and;”>
      <Apply FunctionId=”&or;”>
        <Apply FunctionId=”&string-is-in;”>
          <AttributeValue DataType=”&string;”>Alice</AttributeValue>
          <SubjectAttributeDesignator AttributeId=”&subject-id;”
DataType=”&string;”/>
        </Apply>
        <Apply FunctionId=”&string-is-in;”>
          <AttributeValue DataType=”&string;”>Bob</AttributeValue>
          <SubjectAttributeDesignator AttributeId=”&subject-id;”
DataType=”&string;”/>
        </Apply>
      </Apply>
      <Apply FunctionId=”&string-is-in;”>
        <AttributeValue DataType=”&string;”>H</AttributeValue>
        <ResourceAttributeDesignator AttributeId=”&resource-id;”
DataType=”&string;”/>
      </Apply>
    </Condition>
  </Rule>
  <Rule RuleId=”File:I” Effect=”Permit”>
    <Condition FunctionId=”&and;”>
      <Apply FunctionId=”&string-is-in;”>
        <AttributeValue DataType=”&string;”>Bob</AttributeValue>
        <SubjectAttributeDesignator AttributeId=”&subject-id;”
DataType=”&string;”/>
      </Apply>
      <Apply FunctionId=”&string-is-in;”>
        <AttributeValue DataType=”&string;”>I</AttributeValue>
        <ResourceAttributeDesignator AttributeId=”&resource-id;”
DataType=”&string;”/>
      </Apply>
    </Condition>
  </Rule>
  <Rule RuleId=”File:J” Effect=”Permit”>
    <Condition FunctionId=”&and;”>
      <Apply FunctionId=”&string-is-in;”>
        <AttributeValue DataType=”&string;”>Alice</AttributeValue>
        <SubjectAttributeDesignator AttributeId=”&subject-id;”
DataType=”&string;”/>
      </Apply>
      <Apply FunctionId=”&string-is-in;”>
        <AttributeValue DataType=”&string;”>G</AttributeValue>
        <ResourceAttributeDesignator AttributeId=”&resource-id;”
DataType=”&string;”/>
      </Apply>
    </Condition>
  </Rule>
</Policy>
```

There are other ways this policy could be expressed, but the important point is that, with this solution, each file and directory under directory F must be mentioned specifically in the policy. If new directories and files are created in the future under directory F, then new predicates will need to be supplied in order for clients to have access to them.

4.2 Solution 2: resource-ancestor and resource-parent attributes

This solution uses two new XACML AttributeIds defined in the XACML Profile for Hierarchical Resources:

urn:oasis:names:tc:xacml:2.0:resource:resource-ancestor

urn:oasis:names:tc:xacml:2.0:resource:resource-parent

The “resource-ancestor” AttributeId includes all ancestors of the specific node to which access is requested (i.e. the Attribute with AttributeId of “urn:oasis:names:tc:xacml:2.0:resource:resource-id”). The “resource-parent” AttributeId includes the parents[1] of the “resource-id” node.

Using these two new AttributeId values allows the policy to be stated more compactly. One way of stating the policy using these AttributeId values is as follows.

```
<Policy PolicyId=”PortalPolicy” RuleCombiningAlgId=”&permit-overrides;”>
  <Target>
    <Actions>
      <Action>
        <ActionMatch MatchId=”&string-is-in;”>
```

[1] “resource-parent” will include more than a single “parent” node only if the “resource-id” node is in a hierarchy that is in the form of a “forest” rather than a “tree”.

```
          <AttributeValue DataType=”&string;”>read</AttributeValue>
          <ActionAttributeDesignator AttributeId=”&action-id;”
DataType=”&string;”/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Rule RuleId=”Descendants:of:F” Effect=”Permit”>
    <Condition FunctionId=”&and;”>
      <Apply FunctionId=”&string-is-in;”>
        <AttributeValue DataType=”&string;”>Alice</AttributeValue>
        <SubjectAttributeDesignator AttributeId=”&subject-id;”
DataType=”&string;”/>
      </Apply>
      <Apply FunctionId=”&string-is-in;”>
        <AttributeValue DataType=”&string;”>F</AttributeValue>
        <ResourceAttributeDesignator AttributeId=”&resource-ancestor;”
DataType=”&string;”/>
      </Apply>
      <Apply FunctionId=”&not;”>
        <Apply FunctionId=”&string-is-in;”>
          <AttributeValue DataType=”&string;”>I</AttributeValue>
          <ResourceAttributeDesignator AttributeId=”&resource-id;”
DataType=”&string;”/>
        </Apply>
      </Apply>
    </Condition>
  </Rule>
  <Rule RuleId=”Children:of:G” Effect=”Permit”>
    <Condition FunctionId=”&and;”>
      <Apply FunctionId=”&string-is-in;”>
        <AttributeValue DataType=”&string;”>Bob</AttributeValue>
        <SubjectAttributeDesignator AttributeId=”&subject-id;”
DataType=”&string;”/>
      </Apply>
      <Apply FunctionId=”&string-is-in;”>
        <AttributeValue DataType=”&string;”>G</AttributeValue>
        <ResourceAttributeDesignator AttributeId=”&resource-parent;”
DataType=”&string;”/>
      </Apply>
    </Condition>
  </Rule>
<Policy>
```

In order to use this method, each file and directory MUST have a unique name. If directory F had a sub-directory also named F, for example, a request for an immediate child of the higher directory “F” would get the same privileges as a request for an immediate child of the lower directory “F”. One way to ensure that each node in the hierarchy has a unique name is to use the URI representation for nodes described in the *XACML Profile for Hierarchical Resources* [HIER].

Please note that this method may grant privileges to new directories and files that may be created in the future under the existing directories. For example, if a new directory is created under directory F, then this policy will give Alice the right to read anything in that new directory. This may or may not be the intent of the policy writer, so caution is urged.

4.3 Solution 3: anyURI-match function

This solution uses a function defined in XACML 2.0 with the FunctionId “`urn:oasis:names:tc:xacml:2.0:function:anyURI-match”.` This solution requires that the resources be represented using URIs that reflect the position of each resource in the hierarchy. For example, directory F will be identified as “file://example.com/F” and file H will be identified as “file://example.com/F/G/H”. A description of this naming scheme is included in the *XACML Profile for Hierarchical Resources* [HIER].

This solution solves the problem of assigning unique names to each file and directory, as well as allowing use of the “anyURI-match” function. Nodes named using the URI representation may also use the “resource-ancestor” and “resource-parent” AttributeIds described in Solution 2.

```
<Policy PolicyId=”PortalPolicy” RuleCombiningAlgId=”&permit-overrides;”>
  <Target>
    <Actions>
      <Action>
        <ActionMatch MatchId=”&string-is-in;”>
          <AttributeValue DataType=”&string;”>read</AttributeValue>
          <ActionAttributeDesignator AttributeId=”&action-id;”
DataType=”&string;”/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Rule RuleId=”Descendants:of:F” Effect=”Permit”>
    <Condition FunctionId=”&and;”>
      <Apply FunctionId=”&string-is-in;”>
        <AttributeValue DataType=”&string;”>Alice</AttributeValue>
        <SubjectAttributeDesignator AttributeId=”&subject-id;”
DataType=”&string;”/>
      </Apply>
      <Apply FunctionId=”&anyURI-match;”>
        <AttributeValue
DataType=”&string;”>http://www.example.com/F*</AttributeValue>
        <Apply FunctionId=”&string-one-and-only;”>
          <ResourceAttributeDesignator AttributeId=”&resource-id;”
DataType=”&string;”/>
        </Apply>
      </Apply>
      <Apply FunctionId=”&not;”>
        <Apply FunctionId=”&string-is-in;”>
          <AttributeValue DataType=”&string;”>I</AttributeValue>
          <ResourceAttributeDesignator AttributeId=”&resource-id;”
DataType=”&string;”/>
        </Apply>
      </Apply>
    </Condition>
  </Rule>
  <Rule RuleId=”Children:of:G” Effect=”Permit”>
    <Condition FunctionId=”&and;”>
      <Apply FunctionId=”&string-is-in;”>
        <AttributeValue DataType=”&string;”>Bob</AttributeValue>
        <SubjectAttributeDesignator AttributeId=”&subject-id;”
DataType=”&string;”/>
      </Apply>
      <Apply FunctionId=”&anyURI-match;”>
        <AttributeValue
DataType=”&string;”>http://www.example.com/F/G/*</AttributeValue>
        <Apply FunctionId=”&string-one-and-only;”>
          <ResourceAttributeDesignator AttributeId=”&resource-id;”
DataType=”&string;”/>
        </Apply>
      </Apply>
    </Condition>
  </Rule>
<Policy>
```

Please note that this method may grant privileges to new directories and files that may be created in the future under the existing directories. For example, if a new directory is created under directory F, then this policy will give Alice the right to read anything in that new directory. This may or may not be the intent of the policy writer, so caution is urged.

5 References

[HIER] A. Anderson,ed., *XACML Profile for Hierarchical Resources*, Working Draft 04, 3 June 2004, http://www.oasis-open.org/committees/documents.php?wg_abbrev=xacml

A. Revision History

[This appendix is optional, but helpful. It should be removed for specifications that are at OASIS Standard level. Set the number format for the Rev and Date fields as you wish (select the desired string and choose Format>Number Format...); the examples below are user-defined formats.]

Rev	Date	By Whom	What
01	17 June 2004	Anne Anderson	Initial version.

B. Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

This document and the information contained herein is provided on an “AS IS” basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.