XACML Profile for Role Based Access Control (RBAC)

Committee Specification 01, 13 February 2004

Document identifier:

cs-xacml-rbac-profile-01

Location:

http://docs.oasis-open.org/xacml/cs-xacml-rbac-profile-01.pdf

Editor:

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

Abstract:

This specification defines a profile for the use of XACML in expressing policies that use role based access control (RBAC).

Status:

This version of the specification has been approved as an OASIS Committee Specification.

Committee members should send comments on this specification to the xacml@lists.oasis-open.org list. Others should subscribe to and send comments to the xacml-comment@lists.oasis-open.org list. To subscribe, send an email message to xacml-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

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 XACML TC web page (http://www.oasis-open.org/committees/xacml/).

For any errata page for this specification, please refer to the XACML RBAC Profile section of the XACML TC web page (http://www.oasis-open.org/committees/xacml/).

Table of Contents

1 Introduction (non-normative)

This specification defines a profile for the use of the OASIS eXtensible Access Control Markup Language (XACML) [XACML]to meet the requirements for role based access control (RBAC) as specified in [RBAC]. Use of this Profile requires no changes or extensions to standard XACML Versions 1.0 or 1.1.

This specification begins with a non-normative explanation of the building blocks from which the RBAC solution is constructed. A full example illustrates these building blocks. The specification then discusses how these building blocks may be used to implement the various elements of the RBAC model presented in [RBAC]. Finally, the normative section of the specification describes compliant uses of the building blocks in implementing an RBAC solution.

This proposal assumes the reader is somewhat familiar with XACML. A brief overview sufficient to understand these examples is available in [XACMLIntro]. An introduction to the RBAC model is available in [RBACIntro].

1.1 Notation

In order to improve readability, the examples in this profile assume use of the following XML Internal Entity declarations:

```
^lt;!ENTITY xacml "urn:oasis:names:tc:xacml:1.0:">
^lt;!ENTITY xml "http://www.w3.org/2001/XMLSchema#">
^lt;!ENTITY rule-combine
            "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:">
^lt;!ENTITY policy-combine
            "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:">
^lt;!ENTITY function "urn:oasis:names:tc:xacml:1.0:function:">
^lt;!ENTITY subject-category
            "urn:oasis:names:tc:xacml:1.0:subject-category:">
^lt;!ENTITY subject "urn:oasis:names:tc:xacml:1.0:subject:">
^lt;!ENTITY resource "urn:oasis:names:tc:xacml:1.0:resource:">
^lt;!ENTITY action "urn:oasis:names:tc:xacml:1.0:action:">
^lt;!ENTITY environment "urn:oasis:names:tc:xacml:1.0:environment:">
```

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

1.2 Terminology

The key words *must, must not, required, shall, shall not, should, should not, recommended, may,* and *optional* in this document are to be interpreted as described in IETF RFC 2119 [RFC2119].

attribute - In this Profile, the term "attribute" refers to an XACML `<Attribute>`. An XACML `<Attribute>` is an element in an XACML Request having among its components an attribute name identifier, a data type identifier, and an attribute value. Each `<Attribute>` is associated either with one of the subjects (Subject Attribute), the protected resource (Resource Attribute), the action to be taken on the resource (Action Attribute), or the environment of the Request (Environment Attribute). Attributes are referenced in a policy by using an `<AttributeSelector>` (an XPath expression) or one of the following: `<SubjectAttributeDesignator>`, `<ResourceAttributeDesignator>`, `<ActionAttributeDesignator>`, or `<EnvironmentAttributeDesignator>`.

junior role – In a role hierarchy, Role A is *junior* to Role B if Role B inherits all the permissions associated with Role A.

multi-role permissions – a set of permissions for which a user must hold more than one role simultaneously in order to gain access.

PDP - Policy Decision Point. An entity that evaluates an access request against one or more policies to produce an access decision.

permission – the ability or right to perform some action on some resource, possibly only under certain specified conditions.

PPS – Permission <PolicySet>. See *Section 1.4 Policies*.

RBAC – Role based access control. A model for controlling access to resources where permitted actions on resources are identified with roles rather than with individual subject identities.

RPS – Role <PolicySet>. See *Section 1.4 Policies*.

role – A job function within the context of an organization that has associated semantics regarding the authority and responsibility conferred on the user assigned to the role [RBAC].

senior role – In a role hierarchy, Role A is *senior* to Role B if Role A inherits all the permissions associated with Role B.

policy – A set of rules indicating which subjects are permitted to access which resources using which actions under which conditions.

1.3 Role

*I*n this specification, roles are expressed as XACML Subject Attributes. There is one exception: in a Role Assignment <PolicySet> or <Policy>, the role appears as a Resource Attribute. See Section 3: *Assigning and Enabling Role Attributes* for more information.

Role attributes may be expressed in either of two ways, depending on the preferences of the application environment. In some environments there may be a small number of "role attributes", where the name of each such attribute is some name indicating "role", and where the value of each such attribute indicates the name of the role held. For example, in this first type of environment, there may be one "role attribute" having the identifier urn:someapp:attributes:role. The possible roles are values for this one attribute, and might be officer, manager, and employee. This way of expressing roles works best with the XACML way of expressing policies.

Alternatively, in other application environments, there may be a number of different attribute identifiers, each indicating a different role. For example, in this second type of environment, there might be three attribute identifiers: urn:someapp:attributes:officer-role, urn:someapp:attributes:manager-role, and urn:someapp:attributes:employee-role. In this case the value of the attribute may be empty or it may contain various parameters associated with the role. XACML policies can handle roles expressed in this way, but not as naturally as in the first way.

XACML supports multiple subjects per access request, indicating various entities that may be involved in making the request. For example, there is usually a human user who initiates the request, at least indirectly. There are usually one or more applications or code bases that generate the actual low-level request on behalf of the user. There is some computing device on which the application or code base is executing, and this device may have an identity such an IP address. XACML identifies each such Subject with a SubjectCategory xml attribute that indicates the type of subject being described. For example, the human user has a SubjectCategory of &subject-category;access-subject; (this is the default category); the application that generates the access request has a SubjectCategory of &subject-category;codebase; and so on. In this Profile, a role attribute may be associated with any of the categories of subjects involved in making an access request.

1.4 Policies

In this Profile, there are four types of policies.

1. **Role <PolicySet>** or **RPS** : a <PolicySet> that associates holders of a given role attribute with a Permission <PolicySet> that contains the actual permissions associated with the given role. The <Target> element of a Role <PolicySet> limits the applicability of the <PolicySet> to subjects holding the given role attribute. Each Role <PolicySet> references a single corresponding Permission <PolicySet> but does not contain any other <Policy> or <PolicySet> elements.

2. **Permission <PolicySet>** or **PPS**: a <PolicySet> that contains the actual permissions associated with a given role. It contains <Policy> elements and <Rules> that describe the resources and actions that subjects are permitted to access, along with any further conditions on that access, such as time of day. A given Permission <PolicySet> may also contain references to Permission

<PolicySet>s associated with other roles that are *junior* to the given role, thereby allowing the given Permission <PolicySet> to inherit all permissions associated with the role of the referenced Permission <PolicySet>. The <Target> element of a Permission <PolicySet> must not limit the subjects to which the <PolicySet> is applicable.

3. **Separation of Duty <PolicySet>:** a <PolicySet> that defines restrictions on the set of roles that can be exercised by a given Subject. Such a <PolicySet> contains <Policy> and <Rule> elements that specify the role set restrictions. The Separation of Duty <PolicySet> also contains references to all the Role <PolicySet> instances that are subject to Separation of Duty restrictions. Use of a Separation of Duty <PolicySet> is optional.
4. **Role Assignment <Policy>** or **<PolicySet>:** a <Policy> or <PolicySet> that defines which roles can be enabled or assigned to which subjects. It may also specify restrictions on combinations of roles or total number of roles assigned to or enabled for a given subject. This type of policy is used by the entity that assigns role attributes to users or by the entity that enables role attributes during a user's session. Use of a Role Assignment <Policy> or <PolicySet> is optional.

Permission <PolicySet> instances must be stored in the policy repository in such a way that they can never be used as the initial policy for an XACML PDP; Permission <PolicySet> instances must be reachable only through the corresponding Role <PolicySet>. This is because, in order to support hierarchical roles, a Permission <PolicySet> must be applicable to every subject. The Permission <PolicySet> depends on its corresponding Role <PolicySet> to ensure that only subjects holding the corresponding role attribute will gain access to the permissions in the given Permission <PolicySet>.

If a Separation of Duty <PolicySet> is used, then Role <PolicySet> instances also must be stored in the policy repository in such a way that they can never be used as the initial policy for an XACML PDP. In this case, Role <PolicySet> instances must be reachable only through the Separation of Duty <PolicySet>.

Use of separate Role <PolicySet> and Permission <PolicySet> instances allows support for Hierarchical RBAC, where a more *senior* role can acquire the permissions of a more *junior* role. A Permission <PolicySet> that does not reference other Permission <PolicySet> elements could actually be an XACML <Policy> rather than a <PolicySet>. Requiring it to be a <PolicySet>, however, allows its associated role to become part of a role hierarchy at a later time without requiring any change to other policies.

1.5 Multi-Role Permissions

In this Profile, it is possible to express policies where a user must hold several roles simultaneously in order to gain access to certain permissions. For example, changing the care instructions for a hospital patient may require that the Subject performing the action have both the *physician* role and the *staff* role.

These policies may be expressed using a Role <PolicySet> where the <Target> element requires the Subject to have all necessary role attributes. This is done by using a single <Subject> element containing multiple <SubjectMatch> elements. The associated Permission <PolicySet> should specify the permissions associated with Subjects who simultaneously have all the specified roles enabled.

The Permission <PolicySet> associated with a multi-role policy may reference the Permission <PolicySet> instances associated with other roles, and thus may inherit permissions from other roles. The permissions associated with a given multi-role <PolicySet> may also be inherited by another role if the other role includes a reference to the Permission <PolicySet> associated with the multi-role policy in its own Permission <PolicySet>.

2 Example (non-normative)

This section presents a complete example of the types of policies associated with role based access control.

Assume an organization uses two roles, *manager* and *employee*. In this example, they are expressed as two separate values for a single XACML Attribute with AttributeId "`urn:someapp:attributes:role`", referred to from here on as the `role` Attribute. An *employee* has permission to create a purchase order. A *manager* has permission to sign a purchase order, plus any permissions associated with the employee role.

According to this Profile, there will be two Permission `<PolicySet>` instances: one for the *manager* role and one for the *employee* role. The *manager* Permission `<PolicySet>` will give any `Subject` the specific permission to sign a purchase order and will reference the *employee* Permission `<PolicySet>` in order to inherit its permissions. The *employee* Permission `<PolicySet>` will give any `Subject` the permission to create a purchase order.

According to this Profile, there will also be two Role `<PolicySet>` instances: one for the *manager* role and one for the *employee* role. The *manager* Role `<PolicySet>` will contain a `<Target>` requiring that the `Subject` hold a `role` Attribute with a value of `manager`. It will reference the *manager* Permission `<PolicySet>`. The *employee* Role `<PolicySet>` will contain a `<Target>` requiring that the `Subject` hold a `role` Attribute with a value of `employee`. It will reference the *employee* Permission `<PolicySet>`.

The actual XACML policies implementing this example follow. An example of a Role Assignment Policy is included in Section 3: *Assigning and Enabling Role Attributes*. An example of a Separation of Duty `<PolicySet>` is included in the *Separation of Duty* section of Section 4:.*Implementing the RBAC Model*.

2.1 Permission `<PolicySet>` for the *manager* role

The following Permission `<PolicySet>` contains the permissions associated with the *manager* role. Access to this `<PolicySet>` is gained only by reference from the *manager* Role `<PolicySet>`.

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy"
  PolicySetId="PPS:manager:role"
  PolicyCombiningAlgId="&policy-combine;permit-overrides">
 <Target>
  <Subjects><AnySubject/></Subjects>
  <Resources><AnyResource/></Resources>
  <Actions><AnyAction/></Actions>
 </Target>

 <!-- Permissions specifically for the manager role -->
 <Policy PolicyId="Permissions:specifically:for:the:manager:role"
   RuleCombiningAlgId="&rule-combine;permit-overrides">
  <Target>
   <Subjects><AnySubject/></Subjects>
   <Resources><AnyResource/></Resources>
   <Actions><AnyAction/></Actions>
  </Target>

  <!-- Permission to sign a purchase order -->
  <Rule RuleId="Permission:to:sign:a:purchase:order"
    Effect="Permit">
   <Target>
    <Subjects><AnySubject/></Subjects>
    <Resources>
     <Resource>
      <ResourceMatch MatchId="&function;string-match">
       <AttributeValue
DataType="&xml;string">purchase order</AttributeValue>
       <ResourceAttributeDesignator
         AttributeId="&resource;resource-id"
         DataType="&xml;string"/>
      </ResourceMatch>
```

```
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="&function;string-match">
              <AttributeValue
                  DataType="&xml;string">sign</AttributeValue>
              <ActionAttributeDesignator
                  AttributeId="&action;action-id"
                  DataType="&xml;string"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
  </Policy>

  <!-- Include permissions associated with employee role -->
  <PolicySetIdReference>PPS:employee:role</PolicySetIdReference>
</PolicySet>
```

Table 1 Permission <PolicySet> for managers

2.2 Permission `<PolicySet>` for *employee* role

The following Permission *<PolicySet>* contains the permissions associated with the *employee* role. Access to this `<PolicySet>` is gained only by reference from the *employee* Role `<PolicySet>` or by reference from the more senior *manager* Role `<PolicySet>` via the *manager* Permission `<PolicySet>`.

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy"
     PolicySetId="PPS:employee:role"
     PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <Target>
    <Subjects><AnySubject/></Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>

  <!-- Permissions specifically for the employee role -->
  <Policy PolicyId="Permissions:specifically:for:the:employee:role"
      RuleCombiningAlgId="&rule-combine;permit-overrides">
    <Target>
      <Subjects><AnySubject/></Subjects>
      <Resources><AnyResource/></Resources>
      <Actions><AnyAction/></Actions>
    </Target>

    <!-- Permission to create a purchase order -->
    <Rule RuleId="Permission:to:create:a:purchase:order"
        Effect="Permit">
      <Target>
        <Subjects><AnySubject/></Subjects>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="&function;string-match">
              <AttributeValue
DataType="&xml;string">purchase order</AttributeValue>
              <ResourceAttributeDesignator
                  AttributeId="&resource;resource-id"
                  DataType="&xml;string"/>
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="&function;string-match">
              <AttributeValue
                  DataType="&xml;string">create</AttributeValue>
              <ActionAttributeDesignator
                  AttributeId="&action;action-id"
                  DataType="&xml;string"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
  </Policy>
</PolicySet>
```

Table 2 Permission <PolicySet> for employees

2.3 Role `<PolicySet>` for the *manager* role

The following Role `<PolicySet>` is applicable, according to its `<Target>`, only to `Subjects` who hold a `role` Attribute with a value of `manager`. The `<PolicySetIdReference>` points to the Permission `<PolicySet>` associated with the *manager* role. That Permission `<PolicySet>` may be viewed above.

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy"
    PolicySetId="RPS:manager:role"
    PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="&function;string-equal">
          <AttributeValue
              DataType="&xml;string">manager</AttributeValue>
          <SubjectAttributeDesignator
              AttributeId="urn:someapp:attributes:role"
              DataType="&xml;string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>

  <!-- Use permissions associated with the manager role -->
  <PolicySetIdReference>PPS:manager:role</PolicySetIdReference>
</PolicySet>
```

Table 3 Role <PolicySet> for managers

2.4 Role `<PolicySet>` for *employee* role

The following Role `<PolicySet>` is applicable, according to its `<Target>`, only to `Subjects` who hold a `role` Attribute with a value of `employee`. The `<PolicySetIdReference>` points to the Permission `<PolicySet>` associated with the *employee* role. That Permission `<PolicySet>` may be viewed above.

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy"
    PolicySetId="RPS:employee:role"
    PolicyCombiningAlgId="&policy-combine;permit-overrides">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="&function;string-equal">
          <AttributeValue
              DataType="&xml;string">employee</AttributeValue>
          <SubjectAttributeDesignator
              AttributeId="urn:someapp:attributes:role"
              DataType="&xml;string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>

  <!-- Use permissions associated with the employee role -->
  <PolicySetIdReference>PPS:employee:role</PolicySetIdReference>
</PolicySet>
```

Table 4 Role <PolicySet> for employees

3 Assigning and Enabling Role Attributes (non-normative)

The assignment of various role attributes to users and the enabling of those attributes within a session are outside the scope of the XACML PDP. There must be one or more separate entities defined to perform these functions. This Profile assumes that the presence in the XACML Request Context of a role attribute for a given user (`Subject`) is a valid assignment at the time the access decision is requested

Role assignment entities may, however, use an XACML Role Assignment `<Policy>` or `<PolicySet>` to determine which users are allowed to have various role attributes enabled, and under what conditions. These Role Assignment policies are a different set from the Role `<PolicySet>` and Permission `<PolicySet>` instances used to determine the access permissions associated with each role. Role Assignment policies are to be used only when the XACML Request comes from a role assignment entity.

The following example illustrates a Role Assignment `<Policy>`. It contains two XACML `<Rule>` elements. The first `<Rule>` states that `Anne` and `Seth` and `Yassir` are allowed to have the `employee` role enabled between the hours of 9am and 5pm. The second `<Rule>` states that `Steve` is allowed to have the `manager` role enabled.

```
<Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
    PolicyId="Role:Assignment:Policy"
    RuleCombiningAlgId="&rule-combine;permit-overrides">
<Target>
  <Subjects><AnySubject/></Subjects>
  <Resources><AnyResource/></Resources>
  <Actions><AnyAction/></Actions>
</Target>
```

```
<!--  Employee role requirements rule -->
<Rule RuleId="employee:role:requirements" Effect="Permit">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="&function;string-equal">
          <AttributeValue
              DataType="&xml;string">Seth</AttributeValue>
          <SubjectAttributeDesignator
              AttributeId="&subject;subject-id"
              DataType="&xml;string"/>
        </SubjectMatch>
      </Subject>
      <Subject>
        <SubjectMatch MatchId="&function;string-equal">
          <AttributeValue
              DataType="&xml;string">Anne</AttributeValue>
          <SubjectAttributeDesignator
              AttributeId="&subject;subject-id"
              DataType="&xml;string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="&function;string-equal">
          <AttributeValue
              DataType="&xml;string">employee</AttributeValue>
          <ResourceAttributeDesignator
              AttributeId="urn:someapp:attributes:role"
              DataType="&xml;string"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="&function;string-equal">
          <AttributeValue
              DataType="&xml;string">enable</AttributeValue>
          <ActionAttributeDesignator
              AttributeId="&action;action-id"
              DataType="&xml;string"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Condition FunctionId="&function;and">
    <Apply FunctionId="&function;time-greater-than-or-equal">
      <Apply FunctionId="&function;time-one-and-only">
        <EnvironmentAttributeDesignator
            AttributeId="&environment;current-time"
            DataType="&xml;time"/>
      </Apply>
      <AttributeValue
         DataType="&xml;time">9h</AttributeValue>
    </Apply>
    <Apply FunctionId="&function;time-less-than-or-equal">
      <Apply FunctionId="&function;time-one-and-only">
        <EnvironmentAttributeDesignator
            AttributeId="&environment;current-time"
            DataType="&xml;time"/>
      </Apply>
      <AttributeValue
          DataType="&xml;time">17h</AttributeValue>
   </Apply>
  </Condition>
</Rule>
```

```
    <!-- Manager role requirements rule -->
    <Rule RuleId="manager:role:requirements" Effect="Permit">
      <Target>
        <Subjects>
          <Subject>
            <SubjectMatch MatchId="&function;string-equal">
              <AttributeValue
                   DataType="&xml;string">Steve</AttributeValue>
              <SubjectAttributeDesignator
                   AttributeId="&subject;subject-id"
                   DataType="&xml;string"/>
            </SubjectMatch>
          </Subject>
        </Subjects>
        <Resources>
          <Resource>
            <ResourceMatch MatchId="&function;string-equal">
              <AttributeValue
                  DataType="&xml;string">manager</AttributeValue>
              <ResourceAttributeDesignator
                  AttributeId="urn:someapp:attributes:role"
                  DataType="&xml;string"/>
            </ResourceMatch>
          </Resource>
        </Resources>
        <Actions>
          <Action>
            <ActionMatch MatchId="&function;string-equal">
              <AttributeValue
                  DataType="&xml;string">enable</AttributeValue>
              <ActionAttributeDesignator
                  AttributeId="&action;action-id"
                  DataType="&xml;string"/>
            </ActionMatch>
          </Action>
        </Actions>
      </Target>
    </Rule>
    </Policy>
```

Table 5 Role Assignment <Policy> Example

This policy would be consulted by the entity that makes `role` attributes available for use within a user's session (and thus eligible for being included in an XACML Request Context).

4 Implementing the RBAC Model (non-normative)

The following sections describe how to use XACML policies to implement various components of the RBAC model as described in [RBAC].

4.1 Core RBAC

Core RBAC, as defined in [RBAC], includes the following five basic data elements:

1. **Users**
2. **Roles**
3. **Objects**
4. **Operations**
5. **Permissions**

Users are implemented using XACML `Subjects`. Any of the XACML `SubjectCategory` values may be used, as appropriate.

Roles are expressed using one or more XACML Subject Attributes. The set of roles is very application- and policy domain-specific, and it is very important that different uses of roles not be confused. For these reasons, XACML is not attempting to define any standard set of roles. It is recommended that each application or policy domain agree on and publish a unique set of `AttributeId` values, `DataType` values, and `<AttributeValue>` values that will be used for the various roles relevant to that domain.

Objects are expressed using XACML `Resources`.

Operations are expressed using XACML `Actions`.

Permissions are expressed using XACML Role `<PolicySet>` and Permission `<PolicySet>` instances as described in previous sections.

Core RBAC requires support for multiple users per role, multiple roles per user, multiple permissions per role, and multiple roles per permission. Each of these requirements can be satisfied by XACML policies based on this Profile as follows. Note, however, that the actual assignment of roles to users is outside the scope of the XACML PDP. For more information see Section 3: *Assigning and Enabling Role Attributes*.

XACML allows multiple Subjects to be associated with a given role attribute. XACML Role `<PolicySet>`s defined in terms of possession of a particular role `<Attribute>` and `<AttributeValue>` will apply to any requesting user for which that role `<Attribute>` and `<AttributeValue>` are in the XACML Request Context.

XACML allows multiple role attributes to be associated with a given `Subject`. If a `Subject` has multiple roles enabled, then any Role `<PolicySet>` instance applying to any of those roles may be evaluated, and the permissions in the corresponding Permission `<PolicySet>` will be permitted. As described in the *Policies* Section, it is even possible to define policies that require a given `Subject` to have multiple role attributes enabled at the same time. In this case, the permissions associated with the multiple-role requirement will apply only to a `Subject` having all the necessary role attributes at the time an XACML Request Context is presented to the PDP for evaluation.

The Permission `<PolicySet>` associated with a given role may allow access to multiple resources using multiple actions. XACML has a rich set of constructs for composing permissions, so there are multiple ways in which multi-permission roles may be expressed. Any *Role A* may be associated with a Permission `<PolicySet>` *B* by including a `<PolicySetIdReference>` to Permission `<PolicySet>` *B* in the Permission `<PolicySet>` associated with the *Role A*. In this way, the same set of permissions may be associated with more than one role.

In addition to the basic Core RBAC requirements, XACML policies using this Profile can also express arbitrary conditions on the application of particular permissions associated with a role. Such conditions might include limiting the permissions to a given time period during the day, or limiting the permissions to role holders who also possess some other attribute, whether it is a role attribute or not.

4.2 Hierarchical RBAC

Hierarchical RBAC, as defined in [RBAC], expands Core RBAC with the ability to define inheritance relations between roles. For example, *Role A* may be defined to inherit all permissions associated with *Role B*. In this case, *Role A* is considered to be *senior* to *Role B* in the role hierarchy. If *Role B* in turn inherits permissions associated with *Role C*, then *Role A* will also inherit those permissions by virtue of being senior to *Role B*.

XACML policies using this Profile can implement role inheritance by including a `<PolicySetIdReference>` to the Permission `<PolicySet>` associated with one role inside the Permission `<PolicySet>` associated with another role. The role that includes the `<PolicySetIdReference>` will then inherit the permissions associated with the referenced role.

This Profile structures policies in such a way that inheritance properties may be added to a role at any time without requiring changes to `<PolicySet>` instances associated with any other roles. An organization may not initially use role hierarchies, but may later decide to make use of this functionality without having to rewrite existing policies.

4.3 Separation of Duty

Separation of Duty is a way of avoiding conflicts of interest associated with conflicting roles: a user with one role attribute is not allowed to have some other, conflicting role attribute. S*tatic* Separation of Duty (SSD) relations reduce the number of potential permissions that can be made available to a user by placing constraints on the users that can be assigned to a set of roles. *Dynamic* Separation of Duty (DSD) relations, like SSD relations, are intended to limit the permissions that are available to a user. However DSD relations differ from SSD relations by the context in which these limitations are imposed: they limit the entire space of role attributes that may be associated with a user.

XACML can be used to handle the requirements of Separation of Duty in a number of ways. This Profile recommends use of a Separation of Duty `<PolicySet>` or a Policy Assignment `<PolicySet>`.

Separation of Duty <PolicySet>

A Separation of Duty `<PolicySet>` prevents a user who possesses conflicting role attributes from gaining any access to resources. It acts as a gatekeeper to all the other Role `<PolicySet>` and Permission `<PolicySet>` instances. An example of a Separation of Duty `<PolicySet>` follows. This `<PolicySet>` states that a user may not hold both the *employee* and *contractor* roles at the time an access is requested.

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy"
    PolicySetId="Separation:of:Duty:PolicySet"
    PolicyCombiningAlgId="&policy-combine;deny-overrides">
  <Target>
    <Subjects><AnySubject/></Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>

  <!-- Disallow simultaneous contractor and employee roles -->
  <Policy PolicyId="contractor:AND:employee:disallowed"
      RuleCombiningAlgId="&rule-combine;deny-overrides">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="&function;string-equal">
            <AttributeValue
                DataType="&xml;string">employee</AttributeValue>
            <SubjectAttributeDesignator
                AttributeId="urn:someapp:attributes:role"
                DataType="&xml;string"/>
          </SubjectMatch>
          <SubjectMatch MatchId="&function;string-equal">
            <AttributeValue
                DataType="&xml;string">contractor</AttributeValue>
            <SubjectAttributeDesignator
                AttributeId="urn:someapp:attributes:role"
                DataType="&xml;string"/>
          </SubjectMatch>
        </Subject>
      </Subjects>
      <Resources><AnyResource/></Resources>
      <Actions><AnyAction/></Actions>
    </Target>
    <Rule RuleId="Deny:target:role:combination" Effect="Deny"/>
  </Policy>

  <!-- Reference the Role PolicySets that are subject
       to separation  of duty -->
  <PolicySetIdReference>RPS:employee:role</PolicySetIdReference>
  <PolicySetIdReference>RPS:contractor:role</PolicySetIdReference>
  <PolicySetIdReference>RPS:manager:role</PolicySetIdReference>
</PolicySet>
```

Table 6 Separation of Duty <PolicySet> Example

The Policy or Policies that specify the role restrictions in a Separation of Duty `<PolicySet>` can make use of all the expressiveness of XACML. Restrictions can be placed on the total number of roles held at once, on particular combinations of roles, or on various combinations of conditions.

Role Assignment <PolicySet>

In some environments, it is desirable to prevent a user from being associated with conflicting roles in the first place. Since an XACML PDP does not assign attributes to users, an XACML PDP will not by itself prevent assignment of conflicting role attributes to a user. The entity that performs role assignment or role enablement, however, may make use of a Role Assignment `<PolicySet>` that contains Separation of Duty restrictions.

The following example illustrates an XACML <Rule> that can be included in a Role Assignment `<PolicySet>` implementing a Separation of Duty restriction. It allows *Seth* or *Anne* to enable any two out of the set of possible role attributes:

```
<Rule RuleId="Permission:to:hold:employee:role" Effect="Permit">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="&function;string-equal">
          <AttributeValue
              DataType="&xml;string">Seth</AttributeValue>
          <SubjectAttributeDesignator
              AttributeId="&subject;subject-id"
              DataType="&xml;string"/>
        </SubjectMatch>
      </Subject>
      <Subject>
        <SubjectMatch MatchId="&function;string-equal">
          <AttributeValue
              DataType="&xml;string">Anne</AttributeValue>
          <SubjectAttributeDesignator
              AttributeId="&subject;subject-id"
              DataType="&xml;string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources><AnyResource/></Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="&function;string-equal">
```

```
          <AttributeValue
              DataType="&xml;string">enable</AttributeValue>
          <ActionAttributeDesignator
              AttributeId="&action;action-id"
              DataType="&xml;string"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Target>
  <Condition FunctionId="&function;integer-less-than-or-equal">
    <Apply FunctionId="&function;string-bag-size">
      <ResourceAttributeDesignator
          AttributeId="urn:someapp:attributes:role"
          DataType="&xml;string"/>
    </Apply>
    <AttributeValue
          DataType="&xml;string">2</AttributeValue>
  </Condition>
</Rule>
```

Table 7 Separation of Duty <Rule> Example

Again, the full expressiveness of XACML may be used in specifying role assignment restrictions. Restrictions may be placed on assignment or enablement of particular combinations of roles, on the total number of roles assigned or enabled, or on arbitrary other role assignment or enablement conditions. See Section 3: *Assigning and Enabling Role Attributes* for more information about use of Role Assignment `<PolicySet>`s.

5 Profile (normative)

Roles SHALL be expressed using one or more XACML Attributes. Each application domain using this Profile for role based access control SHALL define or agree upon one or more AttributeId values to be used for role attributes. Each such AttributeId value SHALL be associated with a set of permitted values and their DataTypes. Each permitted value for such an AttributeId SHALL have well-defined semantics for the use of the corresponding value in policies.

5.1 Role Assignment or Enablement

The system entity or entities responsible for issuing role attributes to users and for enabling those attributes for use during a given session MAY use an XACML Role Assignment `<Policy>` or `<PolicySet>` to determine which users are allowed to enable which roles and under which conditions.

5.2 Access Control

Role based access control SHALL be implemented using three types of `<PolicySet>` elements, each with specific functions and requirements as follows. System entities that control access to resources SHALL use XACML Role `<PolicySet>` and Permission `<PolicySet>` policies. Such entities MAY use an XACML Separation of Duty `<PolicySet>`.

For each role, one Role `<PolicySet>` SHALL be defined. Such a `<PolicySet>` SHALL contain a `<Target>` element making the `<PolicySet>` applicable only to holders of the XACML `AttributeId` and `<AttributeValue>` associated with the given role; the `<Target>` element SHALL be applicable to any `Resource` and any `Action`. Each Role `<PolicySet>` SHALL contain a single `<PolicySetIdReference>` element that references the unique Permission `<PolicySet>` associated with the role. The Role `<PolicySet>` SHALL NOT contain any other `<Policy>`, `<PolicySet>`, `<PolicyIdReference>`, or `<PolicySetIdReference>` elements.

For each role, one Permission `<PolicySet>` SHALL be defined. Such a `<PolicySet>` SHALL contain `<Policy>` and `<Rule>` elements that specify the types of access permitted to holders of the Attribute associated with the given role. The `<Target>` of the `<PolicySet>` and its included or referenced `<PolicySet>`, `<Policy>`, and `<Rule>` elements SHALL NOT limit the subjects to which the Permission `<PolicySet>` is applicable; that is, the `<Subjects>` element of each `<Target>` element shall contain an `<AnySubject/>` element.

If a given role inherits permissions from one or more other roles, then the Permission `<PolicySet>` for the given role SHALL include a `<PolicySetIdReference>` element for each other role. Each such `<PolicySetIdReference>` shall reference the Permission `<PolicySet>` associated with the other role from which the given role inherits.

The organization of any repository used for policies and the configuration of the PDP SHALL ensure that the PDP can never use a Permission `<PolicySet>` as the PDP's initial policy.

If a Static Separation of Duty `<PolicySet>` is used, then the organization of any repository used for policies and the configuration of the PDP SHALL ensure that the PDP can never use a Role `<PolicySet>` or Permission `<PolicySet>` as the PDP's initial policy.

6 References

6.1 Normative References

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, IETF RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[XACML] T. Moses, ed., *OASIS eXtensible Access Control Markup Language (XACML) Version 1.1*,http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf, Committee Specification, 24 July 2003.

6.2 Non-normative References

[RBAC] NIST, *Role Based Access Control*,http://csrc.nist.gov/rbac/rbac-std-ncits.pdf, Proposed ANSI Standard, BSR INCITS 359, 4/4/2003.

[RBACIntro] D. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli, *Proposed NIST Standard for Role-Based Access Control*, http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf, ACM Transaction on Information and System Security, Vol. 4, No. 3, August 2001, pages 224-274.

[XACMLIntro] A Brief Introduction to XACML, http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html, 14 March 2003.

A. Acknowledgments

The editor would like to acknowledge the contributions of the OASIS XACML Technical Committee, whose voting members at the time of publication were:

- *Frank Siebenlist, Argonne National Laboratory*
- *Daniel Engovatov, BEA Systems, Inc.*
- *Hal Lockhart, BEA Systems, Inc.*
- *Tim Moses, Entrust*
- *Maryann Hondo, IBM*
- *Michiharu Kudo, IBM*
- *Michael McIntosh, IBM*
- *Anthony Nadalin, IBM*
- *Rebekah Lepro, NASA*
- *Steve Anderson, OpenNetwork*
- *Simon Godik, Overxeer*
- *Bill Parducci, Overxeer*
- *Anne Anderson, Sun Microsystems*
- *Seth Proctor, Sun Microsystems*
- *Polar Humenn, Syracuse University*

In addition, the following people made contributions to this specification:

- *Ravi Sandhu, George Mason Univ.*
- *John Barkley, NIST*
- *Ramaswamy Chandramouli, NIST*
- *David Ferraiolo, NIST*
- *Rick Kuhn, NIST*
- *Serban Gavrila, VDG Inc.*

B. Revision History

Rev	Date	By Whom	What
01	12 Feb 2004	Anne Anderson	Document in Committee Specification format created from the Working Draft at http://www.oasis-open.org/committees/download.php/2405/wd-xacml-rbac-profile-01.doc

C. 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.