XACML v3.0 Core and Hierarchical Role Based Access Control (RBAC) Profile Version 1.0

Committee Draft 01

16 April 2009

Specification URIs:
This Version:

Previous Version:
N/A

Latest Version:
http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-en.html
http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-en.doc (Authoritative)

Technical Committee:
OASIS eXtensible Access Control Markup Language (XACML) TC

Chair(s):
Bill Parducci, <bill@parducci.net>
Hal Lockhart, Oracle <hal.lockhart@oracle.com>

Editor(s):
Erik Rissanen, Axiomatics AB <erik@axiomatics.com>

Related work:
This specification replaces or supercedes:
- Core and hierarchical role based access control (RBAC) profile of XACML v2.0

This specification is related to:
- eXtensible Access Control Markup Language (XACML) Version 3.0, WD 11

Declared XML Namespace(s):
None

Abstract:
This specification defines a profile for the use of XACML in expressing policies that use role based access control (RBAC). It extends the XACML Profile for RBAC Version 1.0 to include a recommended AttributeId for roles, but reduces the scope to address only “core” and “hierarchical” RBAC. This specification has also been updated to apply to XACML 3.0.

Status:
This document was last revised or approved by the eXtensible Access Control Markup Language (XACML) TC on the above date. The level of approval is also listed above. Check the “Latest
Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at http://www.oasis-open.org/committees/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 Technical Committee web page http://www.oasis-open.org/committees/xacml/ipr.php.

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/xacml/.
Notices

Copyright © OASIS® 2009. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the “OASIS IPR Policy”). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

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 OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

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' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on 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 implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names “OASIS” and “XACML” are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.
# Table of Contents

1. Introduction .......................................................................................................................... 5  
   1.1 Glossary .......................................................................................................................... 5  
   1.2 XML Entity Declarations ............................................................................................... 6  
   1.3 Terminology .................................................................................................................... 6  
   1.4 Normative References ..................................................................................................... 6  
   1.5 Non-Normative References ............................................................................................. 6  
   1.6 Scope .................................................................................................................................. 6  
   1.7 Role .................................................................................................................................... 7  
   1.8 Policies .............................................................................................................................. 7  
   1.9 Multi-Role Permissions ...................................................................................................... 8  

2. Example ................................................................................................................................. 9  
   2.1 Permission <PolicySet> for the manager role .................................................................. 9  
   2.2 Permission <PolicySet> for employee role .................................................................... 10  
   2.3 Role <PolicySet> for the manager role .......................................................................... 11  
   2.4 Role <PolicySet> for employee role .............................................................................. 11  
   2.5 HasPrivilegesOfRole Policies and Requests ................................................................ 12  

3. Assigning and Enabling Role Attributes ............................................................................... 14  

4. Implementing the RBAC Model ............................................................................................ 17  
   4.1 1.1 Core RBAC .................................................................................................................. 17  
   4.2 1.2 Hierarchical RBAC ..................................................................................................... 18  

5. Profile ................................................................................................................................... 19  
   5.1 Roles and Role Attributes ............................................................................................... 19  
   5.2 Role Assignment or Enablement ...................................................................................... 19  
   5.3 Access Control .................................................................................................................. 19  

6. Identifiers ............................................................................................................................... 21  
   6.1 Profile Identifier ................................................................................................................ 21  
   6.2 Role Attribute .................................................................................................................... 21  
   6.3 SubjectCategory ............................................................................................................... 21  
   6.4 Action Attribute Values ................................................................................................... 21  

7. Conformance .......................................................................................................................... 22  
   7.1 As a policy processor ........................................................................................................ 22  
   7.2 As an XACML request generator ..................................................................................... 22  

A. Acknowledgements .............................................................................................................. 23  
B. Revision History .................................................................................................................... 24
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 “core” and “hierarchical” role based access control (RBAC) as specified in [ANSI-RBAC]. Use of this profile requires no changes or extensions to standard XACML Version 3.0. Compared to the Core and hierarchical role based access control (RBAC) profile of XACML v2.0 [RBAC-V2] there are no new functionality, rather the specification has just been updated for XACML 3.0.

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 [ANSI-RBAC]. Finally, the normative section of the specification describes compliant uses of the building blocks in implementing an RBAC solution.

This specification 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 Glossary

HasPrivilegesOfRole policy

An optional type of <Policy> that can be included in a Permission <PolicySet> to allow support queries asking if a subject “has the privileges of” a specific role. See Section 2.5: HasPrivilegesOfRole Policies and Requests.

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.

Permission

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

PPS

Permission <PolicySet>. See Section 1.8: 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.

Role Enablement Authority

An entity that assigns role attributes and values to users or enables role attributes and values during a user’s session.

RPS

Role <PolicySet>. See Section 1.8: 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 [ANSI-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.

1.2 XML Entity Declarations

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

```xml
<!ENTITY xml "http://www.w3.org/2001/XMLSchema#">
<!ENTITY rule-combine "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm">
<!ENTITY policy-combine "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm">
<!ENTITY function "urn:oasis:names:tc:xacml:1.0:function">
<!ENTITY subject-category "urn:oasis:names:tc:xacml:1.0:subject-category">
<!ENTITY subject "urn:oasis:names:tc:xacml:1.0:subject">
<!ENTITY role "urn:oasis:names:tc:xacml:2.0:subject:role">
<!ENTITY roles "urn:example:role-values">
<!ENTITY resource "urn:oasis:names:tc:xacml:1.0:resource">
<!ENTITY action "urn:oasis:names:tc:xacml:1.0:action">
<!ENTITY actions "urn:oasis:names:tc:xacml:2.0:actions">
<!ENTITY environment "urn:oasis:names:tc:xacml:1.0:environment">
```

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

1.3 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 [RFC2119].

1.4 Normative References


1.5 Non-Normative References


1.6 Scope

Role based access control allows policies to be specified in terms of subject roles rather than strictly in terms of individual subject identities. This is important for scalability and manageability of access control systems.
The policies specified in this profile can answer three types of questions:

1. If a subject has roles R₁, R₂, ... Rₙ enabled, can subject X access a given resource using a given action?
2. Is subject X allowed to have role Rᵢ enabled?
3. If a subject has roles R₁, R₂, ... Rₙ enabled, does that mean the subject will have permissions associated with a given role R'? That is, is role R' either equal to or junior to any of roles R₁, R₂, ... Rₙ?

The policies specified in this profile do not answer the question “What set of roles does subject X have?” That question must be handled by a Role Enablement Authority, and not directly by an XACML PDP.

In this profile, four types of policies are specified.

1.8 Policies

1. **Role** <PolicySet> or **RPS**: a <PolicySet> that associates holders of a given role attribute and value 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 associated role attribute and value. Each Role <PolicySet> references a single corresponding Permission <PolicySet> but does not contain or reference any other <PolicySet> 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>, if present, must not limit the subjects to which the <PolicySet> is applicable.

3. Role Assignment <PolicySet> or <Policy>: a <PolicySet> or <Policy> 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 a Role Enablement Authority. Use of a Role Assignment <Policy> or <PolicySet> is optional.

4. HasPrivilegesOfRole <Policy>: a <Policy> in a Permission <PolicySet> that supports requests asking whether a subject has the privileges associated with a given role. If this type of request is to be supported, then a HasPrivilegesOfRole <Policy> must be included in each Permission <PolicySet>. Support for this type of <Policy>, and thus for requests asking whether a subject has the privileges associated with a given role, 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>.

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.

1.9 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 <Attributes> element with the subject attribute category to have all necessary role attributes. This is done by using a single <AllOf> element containing multiple <Match> 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

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 "&role;". The &role; Attribute values corresponding to the two roles are "&roles;employee" and "&roles;manager". 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. The manager role therefore is senior to the employee role, and the employee role is junior to the manager 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 "&roles;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 "&roles;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.

2.1 Permission <PolicySet> for the manager role

The following Permission <PolicySet> contains the permissions associated with the manager role. The PDP’s policy retrieval must be set up such that access to this <PolicySet> is gained only by reference from the manager Role <PolicySet>.

```xml
<PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-05"
  PolicySetId="PPS:manager:role"
  PolicyCombiningAlgId="&policy-combine;permit-overrides">

  <!-- Permissions specifically for the manager role -->
  <Policy PolicyId="Permissions:specifically:for:the:manager:role"
    RuleCombiningAlgId="&rule-combine;permit-overrides">

    <!-- Permission to sign a purchase order -->
    <Rule RuleId="Permission:to:sign:a:purchase:order" Effect="Permit">
      <Target>
        <AnyOf>
          <AllOf>
            <Match MatchId="&function;string-equal">
              <AttributeValue
                DataType="&xml;string">purchase order</AttributeValue>
              <AttributeDesignator
                Category="&category;resource"
                AttributeId="&resource;resource-id"
                DataType="&xml;string"/>
            </Match>
          </AllOf>
        </AnyOf>
      </Target>
    </Rule>
  </Policy>

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

The following Permission <PolicySet> contains the permissions associated with the employee role. The PDP's policy retrieval must be set up such that 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>.

```xml
<PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-05"
  PolicySetId="PPS:employee:role"
  PolicyCombiningAlgId="#policy-combine;permit-overrides">
  <!-- Permissions specifically for the employee role -->
  <Rule RuleId="Permissions:specifically:for:the:employee:role"
    RuleCombiningAlgId="#rule-combine;permit-overrides">
    <!-- Permission to create a purchase order -->
    <Target>
      <AnyOf>
        <AllOf>
          <Match MatchId="#function;string-equal">
            <AttributeValue
              DataType="#xml;string">create</AttributeValue>
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
    <Effect>Permit</Effect>
  </Rule>
</PolicySet>
```

Listing 1 Permission <PolicySet> for managers
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 "&roles;manager". The <PolicySetIdReference> points to the Permission <PolicySet> associated with the manager role. That Permission <PolicySet> may be viewed in Section 2.1: Permission <PolicySet> for the manager role above.

```
Listing 2 Permission <PolicySet> for employees
```

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 "&roles;employee". The <PolicySetIdReference> points to the Permission <PolicySet> associated with the employee role. That Permission <PolicySet> may be viewed in Section 2.2: Permission <PolicySet> for employee role above.

```
Listing 3 Role <PolicySet> for managers
```
Listing 4 Role <PolicySet> for employees

2.5 HasPrivilegesOfRole Policies and Requests

An XACML RBAC system MAY choose to support queries of the form “Does this subject have the privileges of role X?” If so, each Permission <PolicySet> MUST contain a HasPrivilegesOfRole <Policy>.

For the Permission <PolicySet> for managers, the HasPrivilegesOfRole <Policy> would look as follows:

Listing 5 HasPrivilegesOfRole <Policy> for manager role

For the Permission <PolicySet> for employees, the HasPrivilegesOfRole <Policy> would look as follows:

Listing 6 HasPrivilegesOfRole <Policy> for employee role
Listing 6  HasPrivilegesOfRole <Policy> for employee role

A Request asking whether subject Anne has the privileges associated with &roles;manager would look as follows.

Listing 7  Example of HasPrivilegesOfRole Request

Either the <Request> must contain Anne's direct roles (in this case, &roles;employee), or else the PDP's Context Handler must be able to discover them. HasPrivilegesOfRole policies do not do the job of associating roles with subjects. See Section 3: Assigning and Enabling Role Attributes for more information on how roles are associated with subjects.
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, referred to as Role Enablement Authorities, implemented 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.

So where do a subject’s role attributes come from? What does one of these Role Enablement Authorities look like? The answer is implementation dependent, but some possibilities can be suggested.

In some cases, role attributes might come from an identity management service that maintains information about a user, including the subject’s assigned or allowed roles; the identity management service acts as the Role Enablement Authority. This service might store static role attributes in an LDAP directory, and a PDP’s Context Handler might retrieve them from there. Or this service might respond to requests for a subject’s role attributes from a PDP’s Context Handler, where the requests are in the form of SAML Attribute Queries.

Role Enablement Authorities MAY use an XACML Role Assignment <Policy> or <PolicySet> to determine whether a subject is allowed to have a particular role attribute and value enabled. A Role Assignment <Policy> or <PolicySet> answers the question “Is subject X allowed to have role R enabled?” It does not answer the question “Which set of roles is subject X allowed to have enabled?” The Role Enablement Authority must have some way of knowing which role or roles to submit a request for. For example, the Role Enablement Authority might maintain a list of all possible roles, and, when asked for the roles associated with a given subject, make a request against the Role Assignment policies for each candidate role.

In this profile, 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 Enablement Authority. This separation may be managed in various ways, such as by using different PDPs with different policy stores or requiring <Request> elements for role enablement queries to include an <Attributes> element with a Category of “&subject-category;role-enablement-authority”.

There is no fixed form for a Role Assignment <Policy>. The following example illustrates one possible form. It contains two XACML <Rule> elements. The first <Rule> states that Anne and Seth and Yassir are allowed to have the “&roles;employee” role enabled between the hours of 9am and 5pm. The second <Rule> states that Steve is allowed to have the “&roles;manager” role enabled, with no restrictions on time of day.

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-05"
    PolicyId="Role:Assignment:Policy"
    RuleCombiningAlgId="&rule-combine;permit-overrides">
    <!-- Employee role requirements rule -->
    <Rule RuleId="employee:role:requirements" Effect="Permit">
        <Target>
            <AnyOf>
                <AllOf>
                    <Match MatchId="&function;string-equal">
                        <AttributeValue
                            DataType="&xml;string">Seth</AttributeValue>
                    </Match>
                    <AttributeDesignator
                        Category="&subject-category;access-subject"
                        AttributeId="&subject;subject-id">
                        <AttributeValue
                            DataType="&xml;string"/>
                    </AttributeDesignator>
                </AllOf>
            </AnyOf>
        </Target>
    </Rule>
</Policy>
```
</Match>
</AllOf>
<AllOf>
  <Match MatchId="&function;string-equal">
    <AttributeValue
      DataType="&xml;string">
      Anne</AttributeValue>
  </Match>
</AllOf>
</AnyOf>
<AnyOf>
  <AllOf>
    <Match MatchId="&function;anyURI-equal">
      <AttributeValue
        DataType="&xml;anyURI">&roles;employee</AttributeValue>
    </Match>
  </AllOf>
</AnyOf>
<AnyOf>
  <AllOf>
    <Match MatchId="&function;anyURI-equal">
      <AttributeValue
        DataType="&xml;anyURI">&actions;enableRole</AttributeValue>
    </Match>
  </AllOf>
</AnyOf>
</Target>
<Condition>
  <Apply FunctionId="&function;and">
    <Apply FunctionId="&function;time-greater-than-or-equal">
      <Apply FunctionId="&function;time-one-and-only">
        <AttributeDesignator
          Category="&category;environment"
          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">
        <AttributeDesignator
          Category="&category;environment"
          AttributeId="&environment;current-time"
          DataType="&xml;time"/>
      </Apply>
      <AttributeValue
        DataType="&xml;time">17h</AttributeValue>
    </Apply>
  </Apply>
</Condition>
</Rule>
<!-- Manager role requirements rule -->

<Rule RuleId="manager:role:requirements" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="&function;string-equal">
          <AttributeValue
            DataType="&xml;string">Steve</AttributeValue>
          <AttributeDesignator
            Category="&subject-category;access-subject"
            AttributeId="&subject;subject-id"
            DataType="&xml;string"/>
          </Match>
        </AllOf>
      </AnyOf>
      <AnyOf>
        <Match MatchId="&function;anyURI-equal">
          <AttributeValue
            DataType="&xml;anyURI">&roles;:manager</AttributeValue>
          <AttributeDesignator
            Category="&category;resource"
            AttributeId="&role;"
            DataType="&xml;anyURI"/>
        </Match>
      </AnyOf>
      <AnyOf>
        <Match MatchId="&function;anyURI-equal">
          <AttributeValue
            DataType="&xml;anyURI">&actions;enableRole</AttributeValue>
          <AttributeDesignator
            Category="&category;action"
            AttributeId="&action;action-id"
            DataType="&xml;anyURI"/>
        </Match>
      </AnyOf>
    </AnyOf>
  </Target>
</Rule>

Listing 8 Role Assignment <Policy> Example
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 [ANSI-RBAC].

4.1 1.1 Core RBAC

{non-normative}

Core RBAC, as defined in [ANSI-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 attribute Category values which are semantically associated with subjects 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, this profile does not attempt to define any standard set of role values, although this profile does recommend use of a common AttributeId value of "urn:oasis:names:tc:xacml:2.0:subject:role". 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 or role attribute values 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 Section 1.9: Multi-Role Permissions, it is even possible to define policies that require a given Subject to have multiple role attributes or values 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 and values 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 1.2 Hierarchical RBAC

Hierarchical RBAC, as defined in [ANSI-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.
5 Profile

5.1 Roles and Role Attributes

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.

This profile RECOMMENDS use of the "urn:oasis:names:tc:xacml:2.0:subject:role" AttributeId value for all role attributes. Instances of this Attribute SHOULD have a DataType of "http://www.w3.org/2001/XMLSchema#anyURI".

5.2 Role Assignment or Enablement

A Role Enablement Authority, responsible for assigning roles to users and for enabling roles for use within a user's session, MAY use an XACML Role Assignment <Policy> or <PolicySet> to determine which users are allowed to enable which roles and under which conditions. There is no prescribed form for a Role Assignment <Policy> or <PolicySet>. It is RECOMMENDED that roles in a Role Assignment <Policy> or <PolicySet> be expressed as Resource Attributes, where the AttributeId is &role; and the <AttributeValue> is the URI for the relevant role value. It is RECOMMENDED that the action of assigning or enabling a role be expressed as an Action Attribute, where the AttributeId is &action;action-id, the DataType is &xml;anyURI, and the <AttributeValue> is &actions;enableRole.

5.3 Access Control

Role based access control SHALL be implemented using two types of <PolicySet>s: Role <PolicySet>, Permission <PolicySet>. The specific functions and requirements of these two types of <PolicySet>s are as follows.

For each role, one Role <PolicySet> SHALL be defined. Such a <PolicySet> SHALL contain a <Target> element that makes the <PolicySet> applicable only to Subjects having the XACML Attribute associated with the given role; the <Target> element SHALL NOT restrict the Resource, Action, or Environment. 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 Subjects having 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.

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

A Permission <PolicySet> MAY include a HasPrivilegesOfRole <Policy>. Such a <Policy> SHALL have a <Rule> element with an effect of "Permit". This Rule SHALL permit any Subject to perform an Action with an Attribute having an AttributeId of &action;action-id, a DataType of &xml;anyURI, and an <AttributeValue> having a value of &actions;hasPrivilegesOfRole on a Resource having an...
Attribute that is the **role** to which the Permission `<PolicySet>` applies (for example, an `AttributeId` of `&role;`, a `DataType` of `&xml;anyURI`, and an `<AttributeValue>` whose value is the URI of the specific `role` value). Note that the `role` Attribute, which is a Subject Attribute in a Role `<PolicySet>` `<Target>`, is treated as a Resource Attribute in a HasPrivilegesOfRole `<Policy>`.

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.
6 Identifiers

This profile defines the following URN identifiers.

6.1 Profile Identifier

The following identifier SHALL be used as the identifier for this profile when an identifier in the form of a URI is required.

\[\text{urn:oasis:names:tc:xacml:3.0:profiles:rbac:core-hierarchical}\]

6.2 Role Attribute

The following identifier MAY be used as the AttributeId for role Attributes.

\[\text{urn:oasis:names:tc:xacml:2.0:subject:role}\]

6.3 SubjectCategory

The following identifier MAY be used as the Category for Subject Attributes identifying that a Request is coming from a Role Enablement Authority.

\[\text{urn:oasis:names:tc:xacml:2.0:subject-category:role-enablement-authority}\]

6.4 Action Attribute Values

The following identifier MAY be used as the <AttributeValue> of the &action;action-id Attribute in a HasPrivilegesOfRole <Policy>.

\[\text{urn:oasis:names:tc:xacml:2.0:actions:hasPrivilegesOfRole}\]

The following identifier MAY be used as the <AttributeValue> of the &action;action-id Attribute in a Role Assignment <Policy>.

\[\text{urn:oasis:names:tc:xacml:2.0:actions:enableRole}\]
7 Conformance

An implementation may conform to this profile in one or more of the following ways.

7.1 As a policy processor
An implementation conforms to this specification as a policy processor if it makes use of XACML policies in the manner described in sections 5 and 6.

7.2 As an XACML request generator
An implementation conforms to this specification as an XACML request generator if it produces XACML requests in the manner described in sections 5 and 6.
A. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

**Participants:**
- Anthony Nadalin
- Bill Parducci
- Erik Rissanen
- Hal Lockhart
- Michiharu Kudo
- Michael McIntosh
- Ron Jacobson
- Seth Proctor
- Steve Anderson
- Tim Moses
## B. Revision History

[optional; should not be included in OASIS Standards]

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Editor</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>WD 1</td>
<td>[Rev Date]</td>
<td>Erik Rissanen</td>
<td>Initial update to XACML 3.0.</td>
</tr>
<tr>
<td>WD 2</td>
<td>28 Dec 2007</td>
<td>Erik Rissanen</td>
<td>Update to the current OASIS template.</td>
</tr>
<tr>
<td>WD 3</td>
<td>4 Nov 2008</td>
<td>Erik Rissanen</td>
<td>Fixed typos in the examples.</td>
</tr>
<tr>
<td>WD 4</td>
<td>5 Apr 2009</td>
<td>Erik Rissanen</td>
<td>Editorial cleanups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Added conformance section.</td>
</tr>
</tbody>
</table>