



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

Committee Specification Draft [0405](#) /
Public Review Draft [0304](#)

~~21 July 2011~~

07 August 2014

Specification URIs

This version:

<http://docs.oasis-open.org/xacml/3.0/rbac/v1.0/csprd04/xacml-3.0-rbac-v1.0-csprd04.doc#A>
(Authoritative)
<http://docs.oasis-open.org/xacml/3.0/rbac/v1.0/csprd04/xacml-3.0-rbac-v1.0-csprd04.html>
<http://docs.oasis-open.org/xacml/3.0/rbac/v1.0/csprd04/xacml-3.0-rbac-v1.0-csprd04.pdf>

Previous version:

<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-csprd03-en.doc> (Authoritative)
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-csprd03-en.html>
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-csprd03-en.pdf>

Latest version:

<http://docs.oasis-open.org/xacml/3.0/rbac/v1.0/xacml-3.0-rbac-v1.0.doc#A>
(Authoritative)
<http://docs.oasis-open.org/xacml/3.0/rbac/v1.0/xacml-3.0-rbac-v1.0.html>
<http://docs.oasis-open.org/xacml/3.0/rbac/v1.0/xacml-3.0-rbac-v1.0.pdf>

Technical Committee:

OASIS eXtensible Access Control Markup Language (XACML) TC

Chairs:

Bill Parducci (bill@parducci.net), Individual
Hal Lockhart (hal.lockhart@oracle.com), Oracle

Editor:

Erik Rissanen (erik@axiomantics.com), Axiomatics-AB

Related work:

This specification replaces or supersedes:

- *Core and hierarchical role based access control (RBAC) profile of XACML v2.0*. [Edited by Anne Anderson](#). 1 February 2005. OASIS Standard. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf.

This specification is related to:

- *eXtensible Access Control Markup Language (XACML) Version 3.0*. [Edited by Erik Rissanen](#). Latest version: <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html>.

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 Attribute field for roles, but reduces the scope to address only “core” and “hierarchical” RBAC. This specification has also been updated to apply to XACML ~~3v3.0~~.

Status:

This document was last revised or approved by the OASIS eXtensible Access Control Markup Language (XACML) TC on the above date. The level of approval is also listed above. Check the “Latest 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 <https://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 (<https://www.oasis-open.org/committees/xacml/ipr.php>).

Citation format:

When referencing this specification the following citation format should be used:

[XACML-3.0-RBAC]

XACML v3.0 Core and Hierarchical Role Based Access Control (RBAC) Profile Version 1.0. ~~24 July 2014~~. Edited by Erik Rissanen. 07 August 2014. OASIS Committee Specification Draft ~~0405~~ / Public Review Draft ~~04~~. <http://docs.oasis-open.org/xacml/3.0/rbac/v1.0/csprd04/xacml-3.0-rbac-v1.0-csprd04.html>. Latest version: http://docs.oasis-open.org/xacml/3.0/rbac/v1.0/xacml-3.0-rbac-v1.0.html03_

Notices

Copyright © OASIS Open 2014. 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 <https://www.oasis-open.org/whpolicies-guidelines/trademark.php> for above guidance.

Table of Contents

1	Introduction.....	5
1.1	Background.....	5
1.2	Glossary.....	5
1.3	XML Entity Declarations	6
1.4	Terminology	6
1.5	Normative References	6
1.6	Non-Normative References	6
1.7	Scope.....	6
1.8	Role.....	7
1.9	Policies.....	7
1.10	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	12
2.5	HasPrivilegesOfRole Policies and Requests.....	12
3	Assigning and Enabling Role Attributes	15
4	Implementing the RBAC Model.....	15
4.1	Core RBAC	18
4.2	Hierarchical RBAC	19
5	Profile	20
5.1	Roles and Role Attributes	20
5.2	Role Assignment or Enablement	20
5.3	Access Control.....	20
6	Identifiers	22
6.1	Profile Identifier	22
6.2	Role Attribute	22
6.3	Action Attribute Values	22
7	Conformance	23
7.1	As a policy processor.....	23
7.2	As an XACML request generator.....	23
Appendix A.	Acknowledgments.....	24
Appendix B.	Revision History	25

1 Introduction

1.1 Background

{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 is 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. An introduction to the **RBAC** model is available in [RBACIntro].

1.1.2 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 1.9: 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 1.9: Policies.

Role

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

44 Senior role

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

47 1.21.3 XML Entity Declarations

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

50

```
51 <!ENTITY xml "http://www.w3.org/2001/XMLSchema#">  
52 <!ENTITY rule-combine "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:">  
53 <!ENTITY policy-combine "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:">  
54 <!ENTITY function "urn:oasis:names:tc:xacml:1.0:function:">  
55 <!ENTITY subject-category "urn:oasis:names:tc:xacml:1.0:subject-category:">  
56 <!ENTITY subject "urn:oasis:names:tc:xacml:1.0:subject:">  
57 <!ENTITY role "urn:oasis:names:tc:xacml:2.0:subject:role:">  
58 <!ENTITY roles "urn:example:role-values:">  
59 <!ENTITY resource "urn:oasis:names:tc:xacml:1.0:resource:">  
60 <!ENTITY action "urn:oasis:names:tc:xacml:1.0:action:">  
61 <!ENTITY actions "urn:oasis:names:tc:xacml:2.0:actions:">  
62 <!ENTITY environment "urn:oasis:names:tc:xacml:1.0:environment:">  
63 <!ENTITY category "urn:oasis:names:tc:xacml:3.0:attribute-category:">
```

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

65 1.31.4 Terminology

66 The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD
67 NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described
68 in [RFC2119].

69 1.41.5 Normative References

- 70 [RFC2119] ~~S.~~ Bradner, ~~S.~~, “Key words for use in RFCs to Indicate Requirement Levels,
71 ~~IETF~~, BCP 14, RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.
72 [XACML] ~~OASIS Committee Draft 03~~, *eXtensible Access Control Markup Language*
73 *(XACML) Version 3.0*, ~~11 March 2010~~, 22 January 2014. OASIS Standard.
74 <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html>

75 1.51.6 Non-Normative References

- 76 [ANSI-RBAC] NIST, Role Based Access Control, ANSI INCITS 359-2004,
77 <http://csrc.nist.gov/rbac/>
78 [RBACIntro] D. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli, Proposed
79 NIST Standard for Role-Based Access Control, ACM Transaction on Information
80 and System Security, Vol. 4, No. 3, August 2001, pages 224-274,
81 <http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf>
82 [RBAC-V2] ~~OASIS Standard~~, *Core and hierarchical role based access control (RBAC) profile*
83 *of XACML v2.0*, 1 February 2005, ~~OASIS Standard~~. [http://docs.oasis-](http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf)
84 [open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf](http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf)

85 1.61.7 Scope

86 **Role** based access control allows policies to be specified in terms of subject **roles** rather than strictly in
87 terms of individual subject identities. This is important for scalability and manageability of access control
88 systems.

- 89 | The policies specified in this profile can answer [threetwo](#) types of questions:
- 90 | 1. If a subject has **roles** R1 , R2, ... Rn enabled, can subject X access a given resource using a
- 91 | given action?
- 92 | ~~1. Is subject X allowed to have **role** R_i enabled?~~
- 93 | 2. If a subject has **roles** R1 , R2, ... Rn enabled, does that mean the subject will have **permissions**
- 94 | associated with a given **role** R'? That is, is **role** R' either equal to or junior to any of **roles** R1 ,
- 95 | R2, ... Rn?

96 | The policies specified in this profile do not answer the question “What set of **roles** does subject X have?”

97 | That question must be handled by a **Role Enablement Authority**, and not directly by an XACML PDP.

98 | Such an entity may make use of XACML policies, but will need additional information. See Section [:3:](#)

99 | Assigning and Enabling Role Attributes for more information about **Role Enablement Authorities**.

100 | The policies specified in this profile assume all the **roles** for a given subject have already been enabled at

101 | the time an authorization decision is requested. They do not deal with an environment in which **roles**

102 | must be enabled dynamically based on the resource or actions a subject is attempting to perform. For

103 | this reason, the policies specified in this profile also do not deal with static or dynamic “Separation of

104 | Duty” (see [\[ANSI-RBAC\]](#)). A future profile may address the requirements of this type of environment.

105 | [1.71.8 Role](#)

106 | In this profile, **roles** are expressed as XACML Subject Attributes. [There are two exceptions: in a Role](#)

107 | [Assignment <PolicySet> or <Policy> and There is one exception:](#) in a HasPrivilegesOfRole

108 | <Policy>, the **role** appears as a Resource Attribute. See Section 2.5: HasPrivilegesOfRole Policies

109 | and Requests [and Section : Assigning and Enabling Role Attributes](#) for more information.

110 | **Role** attributes may be expressed in either of two ways, depending on the requirements of the application

111 | environment. In some environments there may be a small number of “**role** attributes”, where the name of

112 | each such attribute is some name indicating “role”, and where the value of each such attribute indicates

113 | the name of the **role** held. For example, in this first type of environment, there may be one “**role** attribute”

114 | having the `AttributeId` “&role;” (this profile recommends use of this identifier). The possible **roles** are

115 | values for this one attribute, and might be “&roles;officer”, “&roles;manager”, and “&roles;employee”. This

116 | way of expressing **roles** works best with the XACML way of expressing policies. This method of

117 | identifying **roles** is also most conducive to interoperability.

118 | Alternatively, in other application environments, there may be a number of different attribute identifiers,

119 | each indicating a different **role**. For example, in this second type of environment, there might be three

120 | attribute identifiers: “urn:someapp:attributes:officer-role”, “urn:someapp:attributes:manager-role”, and

121 | “urn:someapp:attributes:employee-role”. In this case the value of the attribute may be empty or it may

122 | contain various parameters associated with the **role**. XACML policies can handle **roles** expressed in this

123 | way, but not as naturally as in the first way.

124 | XACML supports multiple subjects per access request, indicating various entities that may be involved in

125 | making the request. For example, there is usually a human user who initiates the request, at least

126 | indirectly. There are usually one or more applications or code bases that generate the actual low-level

127 | access request on behalf of the user. There is some computing device on which the application or code

128 | base is executing, and this device may have an identity such an IP address. XACML identifies each such

129 | Subject with a `Category` xml attribute in the <Attributes> element that indicates the type of subject

130 | being described. For example, the human user has a `Category` of &subject-category;access-subject;

131 | the application that generates the access request has a `Category` of &subject-category;codebase and

132 | so on. In this profile, a **role** attribute may be associated with any of the categories of subjects involved in

133 | making an access request.

134 | [1.81.9 Policies](#)

135 | In this profile, [fourthree](#) types of policies are specified.

- 136 | 1. **Role <PolicySet> or RPS**: a <PolicySet> that associates holders of a given **role** attribute
- 137 | and value with a Permission <PolicySet> that contains the actual **permissions** associated with

138 the given **role**. The <Target> element of a Role <PolicySet> limits the applicability of the
139 <PolicySet> to subjects holding the associated **role** attribute and value. Each Role
140 <PolicySet> references a single corresponding Permission <PolicySet> but does not
141 contain or reference any other <Policy> or <PolicySet> elements.

142 2. **Permission <PolicySet> or PPS:** a <PolicySet> that contains the actual **permissions**
143 associated with a given **role**. It contains <PolicySet> and <Policy> elements and <Rules>
144 that describe the resources and actions that subjects are permitted to access, along with any
145 further conditions on that access, such as time of day. A given Permission <PolicySet> may
146 also contain references to Permission <PolicySet>s associated with other **roles** that are junior
147 to the given **role**, thereby allowing the given Permission <PolicySet> to inherit all **permissions**
148 associated with the **role** of the referenced Permission <PolicySet>. The <Target> element of
149 a Permission <PolicySet>, if present, must not limit the subjects to which the <PolicySet> is
150 applicable.

151 ~~1. **Role Assignment <Policy> or <PolicySet>:** a <Policy> or <PolicySet> that defines
152 which **roles** can be enabled or assigned to which subjects. It may also specify restrictions on
153 combinations of **roles** or total number of **roles** assigned to or enabled for a given subject. This
154 type of policy is used by a **Role Enablement Authority**. Use of a Role Assignment <Policy> or
155 <PolicySet> is optional.~~

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

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

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

174 **1.91.10 Multi-Role Permissions**

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

178 These policies may be expressed using a Role <PolicySet> where the <Target> element requires the
179 <Attributes> element with the subject attribute category to have all necessary **role** attributes. This is
180 done by using a single <AllOf> element containing multiple <Match> elements. The associated
181 Permission <PolicySet> should specify the **permissions** associated with Subjects who simultaneously
182 have all the specified **roles** enabled.

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

2 Example

188

189 {non-normative}

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

192 Assume an organization uses two **roles**, manager and employee. In this example, they are expressed
193 as two separate values for a single XACML Attribute with `AttributeId` “&role;”. The &role; Attribute
194 values corresponding to the two **roles** are “&roles;employee” and “&roles;manager”. An employee has
195 **permission** to create a purchase order. A manager has **permission** to sign a purchase order, plus any
196 **permissions** associated with the employee **role**. The manager **role** therefore is senior to the employee
197 **role**, and the employee **role** is junior to the manager **role**.

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

203 According to this profile, there will also be two `Role` <PolicySet> instances: one for the manager **role**
204 and one for the employee **role**. The manager `Role` <PolicySet> will contain a <Target> requiring that
205 the Subject hold a &role; Attribute with a value of “&roles;manager”. It will reference the manager
206 `Permission` <PolicySet>. The employee `Role` <PolicySet> will contain a <Target> requiring that
207 the Subject hold a &role; Attribute with a value of “&roles;employee”. It will reference the employee
208 `Permission` <PolicySet>.

209 The actual XACML policies implementing this example follow. [An example of a Role Assignment Policy is
210 included in Section : Assigning and Enabling Role Attributes.](#)

2.1 Permission <PolicySet> for the manager role

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

215

```
216 <PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"  
217   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
218   xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-  
219   core-v3-schema-wd-17.xsd"  
220   PolicySetId="PPS:manager:role"  
221   Version="1.0"  
222   PolicyCombiningAlgId="&policy-combine;permit-overrides">  
223     <Target/>  
224  
225     <!-- Permissions specifically for the manager role -->  
226     <Policy PolicyId="Permissions:specifically:for:the:manager:role"  
227       Version="1.0"  
228       RuleCombiningAlgId="&rule-combine;permit-overrides">  
229       <Target/>  
230       <!-- Permission to sign a purchase order -->  
231       <Rule RuleId="Permission:to:sign:a:purchase:order" Effect="Permit">  
232         <Target>  
233           <AnyOf>  
234             <AllOf>  
235               <Match MatchId="&function:string-equal">  
236                 <AttributeValue  
237                   DataType="&xml:string">purchase order</AttributeValue>
```

```

238         <AttributeDesignator
239             MustBePresent="false"
240             Category="&category;resource"
241             AttributeId="&resource;resource-id"
242             DataType="&xml;string"/>
243     </Match>
244 </AllOf>
245 </AnyOf>
246 <AnyOf>
247     <AllOf>
248         <Match MatchId="&function;string-equal">
249             <AttributeValue
250                 DataType="&xml;string">sign</AttributeValue>
251             <AttributeDesignator
252                 MustBePresent="false"
253                 Category="&category;action"
254                 AttributeId="&action;action-id"
255                 DataType="&xml;string"/>
256         </Match>
257     </AllOf>
258 </AnyOf>
259 </Target>
260 </Rule>
261 </Policy>
262
263 <!-- Include permissions associated with employee role -->
264 <PolicySetIdReference>PPS:employee:role</PolicySetIdReference>
265 </PolicySet>

```

266 *Listing 1 Permission <PolicySet> for managers*

267 2.2 Permission <PolicySet> for employee role

268 The following Permission <PolicySet> contains the **permissions** associated with the employee **role**.
269 The PDP's policy retrieval must be set up such that access to this <PolicySet> is gained only by
270 reference from the employee Role <PolicySet> or by reference from the more senior manager Role
271 <PolicySet> via the manager Permission <PolicySet>.

272

```

273 <PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
274     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
275     xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-
276 core-v3-schema-wd-17.xsd"
277     PolicySetId="PPS:employee:role"
278     Version="1.0"
279     PolicyCombiningAlgId="&policy-combine;permit-overrides">
280
281     <Target/>
282     <!-- Permissions specifically for the employee role -->
283     <Policy PolicyId="Permissions:specifically:for:the:employee:role"
284         Version="1.0"
285         RuleCombiningAlgId="&rule-combine;permit-overrides">
286     <Target/>
287     <!-- Permission to create a purchase order -->
288     <Rule RuleId="Permission:to:create:a:purchase:order" Effect="Permit">
289     <Target>
290     <AnyOf>
291     <AllOf>
292     <Match MatchId="&function;string-equal">
293     <AttributeValue
294         DataType="&xml;string">purchase order</AttributeValue>
295     <AttributeDesignator
296     MustBePresent="false"

```

```

297         Category="&category;resource"
298         AttributeId="&resource;resource-id"
299         DataType="&xml:string"/>
300     </Match>
301 </AllOf>
302 </AnyOf>
303 <AnyOf>
304     <AllOf>
305         <Match MatchId="&function;string-equal">
306             <AttributeValue
307                 DataType="&xml:string">create</AttributeValue>
308             <AttributeDesignator
309                 MustBePresent="false"
310                 Category="&category;action"
311                 AttributeId="&action;action-id"
312                 DataType="&xml:string"/>
313         </Match>
314     </AllOf>
315 </AnyOf>
316 </Target>
317 </Rule>
318 </Policy>
319 </PolicySet>

```

320 *Listing 2 Permission <PolicySet> for employees*

321 **2.3 Role <PolicySet> for the manager role**

322 The following Role <PolicySet> is applicable, according to its <Target>, only to Subjects who hold a
323 &role; Attribute with a value of "&roles;manager". The <PolicySetIdReference> points to the
324 Permission <PolicySet> associated with the manager **role**. That Permission <PolicySet> may be
325 viewed in Section 2.1: Permission <PolicySet> for the manager **role** above.

326

```

327 <PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
328     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
329     xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-
330 core-v3-schema-wd-17.xsd"
331     PolicySetId="RPS:manager:role"
332     Version="1.0"
333     PolicyCombiningAlgId="&policy-combine;permit-overrides">
334 <Target>
335     <AnyOf>
336         <AllOf>
337             <Match MatchId="&function;anyURI-equal">
338                 <AttributeValue
339                     DataType="&xml:anyURI">&roles;manager</AttributeValue>
340                 <AttributeDesignator
341                     MustBePresent="false"
342                     Category="&subject-category;access-subject"
343                     AttributeId="&role;"
344                     DataType="&xml:anyURI"/>
345             </Match>
346         </AllOf>
347     </AnyOf>
348 </Target>
349
350     <!-- Use permissions associated with the manager role -->
351     <PolicySetIdReference>PPS:manager:role</PolicySetIdReference>
352 </PolicySet>

```

353 *Listing 3 Role <PolicySet> for managers*

354 2.4 Role <PolicySet> for employee role

355 The following Role <PolicySet> is applicable, according to its <Target>, only to Subjects who hold a
356 &role; Attribute with a value of "&roles;employee". The <PolicySetIdReference> points to the
357 Permission <PolicySet> associated with the employee **role**. That Permission <PolicySet> may be
358 viewed in Section 2.2: Permission <PolicySet> for employee **role** above.

359

```
360 <PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"  
361   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
362   xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-  
363 core-v3-schema-wd-17.xsd"  
364   PolicySetId="RPS:employee:role"  
365   Version="1.0"  
366   PolicyCombiningAlgId="&policy-combine;permit-overrides">  
367   <Target>  
368     <AnyOf>  
369       <AllOf>  
370         <Match MatchId="&function;anyURI-equal">  
371           <AttributeValue  
372             DataType="&xml;anyURI">&roles;employee</AttributeValue>  
373           <AttributeDesignator  
374             MustBePresent="false"  
375             Category="&subject-category;access-subject"  
376             AttributeId="&role;"  
377             DataType="&xml;anyURI"/>  
378         </Match>  
379       </AllOf>  
380     </AnyOf>  
381   </Target>  
382  
383   <!-- Use permissions associated with the employee role -->  
384   <PolicySetIdReference>PPS:employee:role</PolicySetIdReference>  
385 </PolicySet>
```

386 *Listing 4 Role <PolicySet> for employees*

387 2.5 HasPrivilegesOfRole Policies and Requests

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

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

393

```
394 <!-- HasPrivilegesOfRole Policy for manager role -->  
395 <Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"  
396   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
397   xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-  
398 core-v3-schema-wd-17.xsd"  
399   PolicyId="Permission:to:have:manager:role:permissions"  
400   Version="1.0"  
401   RuleCombiningAlgId="&rule-combine;permit-overrides">  
402  
403   <Target/>  
404   <!-- Permission to have manager role permissions -->  
405   <Rule RuleId="Permission:to:have:manager:permissions" Effect="Permit">  
406     <Condition>  
407       <Apply FunctionId="&function;and">  
408         <Apply FunctionId="&function;anyURI-is-in">
```

```

409     <AttributeValue
410         DataType="&xml;anyURI">&roles;manager</AttributeValue>
411     <AttributeDesignator
412         MustBePresent="false"
413         Category="&category;resource"
414         AttributeId="&role;"
415         DataType="&xml;anyURI"/>
416 </Apply>
417 <Apply FunctionId="&function;anyURI-is-in">
418     <AttributeValue
419         DataType="&xml;anyURI">&actions;hasPrivilegesofRole</AttributeValue>
420     <AttributeDesignator
421         MustBePresent="false"
422         Category="&category;action"
423         AttributeId="&action;action-id"
424         DataType="&xml;anyURI"/>
425     </Apply>
426 </Apply>
427 </Condition>
428 </Rule>
429 </Policy>

```

430 *Listing 5 HasPrivilegesOfRole <Policy> for manager role*

431

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

434

```

435 <!-- HasPrivilegesOfRole Policy for employee role -->
436 <Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
437     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
438     xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-
439 core-v3-schema-wd-17.xsd"
440     PolicyId="Permission:to:have:employee:role:permissions"
441     Version="1.0"
442     RuleCombiningAlgId="&rule-combine;permit-overrides">
443
444     <Target/>
445     <!-- Permission to have employee role permissions -->
446     <Rule RuleId="Permission:to:have:employee:permissions" Effect="Permit">
447         <Condition>
448             <Apply FunctionId="&function;and">
449                 <Apply FunctionId="&function;anyURI-is-in">
450                     <AttributeValue
451                         DataType="&xml;anyURI">&roles;employee</AttributeValue>
452                     <AttributeDesignator
453                         MustBePresent="false"
454                         Category="&category;resource"
455                         AttributeId="&role;"
456                         DataType="&xml;anyURI"/>
457                     </Apply>
458                 <Apply FunctionId="&function;anyURI-is-in">
459                     <AttributeValue
460                         DataType="&xml;anyURI">&actions;hasPrivilegesofRole</AttributeValue>
461                     <AttributeDesignator
462                         MustBePresent="false"
463                         Category="&category;action"
464                         AttributeId="&action;action-id"
465                         DataType="&xml;anyURI"/>
466                     </Apply>
467                 </Apply>
468             </Condition>

```

```
469     </Rule>
470 </Policy>
```

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

472

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

475

```
476 <Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
477     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
478     xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-
479 core-v3-schema-wd-17.xsd"
480     CombinedDecision="false"
481     ReturnPolicyIdList="false">
482 <Attributes Category="&subject-category;access-subject">
483 <Attribute AttributeId="&subject;subject-id"
484     IncludeInResult="false">
485 <AttributeValue DataType="&xml:string">Anne</AttributeValue>
486 </Attribute>
487 </Attributes>
488 <Attributes Category="&category;resource">
489 <Attribute AttributeId="&role;"
490     IncludeInResult="false">
491 <AttributeValue DataType="&xml:anyURI">&roles;manager</AttributeValue>
492 </Attribute>
493 </Attributes>
494 <Attributes Category="&category;action">
495 <Attribute AttributeId="&action;action-id"
496     IncludeInResult="false">
497 <AttributeValue
498     DataType="&xml:anyURI">&actions;hasPrivilegesOfRole</AttributeValue>
499 </Attribute>
500 </Attributes>
501 </Request>
```

502 *Listing 7 Example of HasPrivilegesOfRole Request*

503

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

3 Assigning and Enabling Role Attributes

508

509 {non-normative}

510 The assignment of various *role* attributes to users and the enabling of those attributes within a session
511 are outside the scope of the XACML PDP. There must be one or more separate entities, referred to a
512 **Role Enablement Authorities**, implemented to perform these functions. This profile assumes that the
513 presence in the XACML Request Context of a *role* attribute for a given user (Subject) is a valid
514 assignment at the time the access decision is requested

515 So where do a subject's *role* attributes come from? What does one of these **Role Enablement**
516 **Authorities** look like? The answer is implementation dependent, but some possibilities can be
517 suggested and this profile prescribes no specific form for them.

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

524 **Role Enablement Authorities** MAY could use an XACML **Role Assignment** `<Policy>` or
525 `<PolicySet>` policies to determine whether a subject is allowed to have a particular *role* attribute and
526 value enabled. A Role Assignment `<Policy>` or `<PolicySet>` answers the question "Is subject X
527 allowed to have role R_i enabled?" It does not answer the question "Which set of roles is subject X
528 allowed to have enabled?" The Role Enablement Authority must have some way of knowing which role
529 or roles to submit a request for. For example, the Role Enablement Authority might maintain a list of
530 all However, there are multiple possible roles, and, when asked for the roles associated with a given
531 subject, make a request against the Role Assignment policies for each candidate role.

532 In this profile, Role Assignment policies are a different set from the Role `<PolicySet>` and Permission
533 <PolicySet> instances used to determine the access permissions associated with each role. Role
534 Assignment policies are to be used only when the XACML Request comes from a Role Enablement
535 Authority. This separation may be managed in various ways, such as by using different PDPs with
536 different policy stores or requiring `<Request>` elements for role enablement queries to include an
537 <Attributes> element with a Category of "&subject-category;role-enablement-authority".

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

543

```
544 <Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"  
545 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
546 xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-  
547 core-v3-schema-wd-17.xsd"  
548 PolicyId="Role:Assignment:Policy"  
549 Version="1.0"  
550 RuleCombiningAlgId="&rule-combine;permit-overrides">  
551 <Target/>
```

553 <!-- Employee role the specific requirements rule -->, so the XACML TC has decided to not standardize
554 any specific form for such policies in this profile.

```
555 <Rule RuleId="employee:role:requirements" Effect="Permit">  
556 <Target>  
557 <AnyOf>  
558 <AllOf>
```

```

559 |         <Match MatchId="&function;string-equal">
560 |             <AttributeValue
561 |                 DataType="&xml;string">Seth</AttributeValue>
562 |             <AttributeDesignator
563 |                 MustBePresent="false"
564 |                 Category="&subject-category;access-subject"
565 |                 AttributeId="&subject;subject-id"
566 |                 DataType="&xml;string"/>
567 |         </Match>
568 |     </AllOf>
569 | <AllOf>
570 |     <Match MatchId="&function;string-equal">
571 |         <AttributeValue
572 |             DataType="&xml;string">Anne</AttributeValue>
573 |         <AttributeDesignator
574 |             MustBePresent="false"
575 |             Category="&subject-category;access-subject"
576 |             AttributeId="&subject;subject-id"
577 |             DataType="&xml;string"/>
578 |     </Match>
579 | </AllOf>
580 | </AnyOf>
581 | <AnyOf>
582 | <AllOf>
583 |     <Match MatchId="&function;anyURI-equal">
584 |         <AttributeValue
585 |             DataType="&xml;anyURI">&roles;employee</AttributeValue>
586 |         <AttributeDesignator
587 |             MustBePresent="false"
588 |             Category="&category;resource"
589 |             AttributeId="&role;"
590 |             DataType="&xml;anyURI"/>
591 |     </Match>
592 | </AllOf>
593 | </AnyOf>
594 | <AnyOf>
595 | <AllOf>
596 |     <Match MatchId="&function;anyURI-equal">
597 |         <AttributeValue
598 |             DataType="&xml;anyURI">&actions;enableRole</AttributeValue>
599 |         <AttributeDesignator
600 |             MustBePresent="false"
601 |             Category="&category;action"
602 |             AttributeId="&action;action-id"
603 |             DataType="&xml;anyURI"/>
604 |     </Match>
605 | </AllOf>
606 | </AnyOf>
607 | </Target>
608 | <Condition>
609 |     <Apply FunctionId="&function;and">
610 |     <Apply FunctionId="&function;time-greater-than-or-equal">
611 |     <Apply FunctionId="&function;time-one-and-only">
612 |         <AttributeDesignator
613 |             MustBePresent="false"
614 |             Category="&category;environment"
615 |             AttributeId="&environment;current-time"
616 |             DataType="&xml;time"/>
617 |     </Apply>
618 |     <AttributeValue
619 |         DataType="&xml;time">9h</AttributeValue>
620 |     </Apply>
621 |     <Apply FunctionId="&function;time-less-than-or-equal">
622 |     <Apply FunctionId="&function;time-one-and-only">

```

```

623 |         <AttributeDesignator
624 |             MustBePresent="false"
625 |             Category="&category;environment"
626 |             AttributeId="&environment;current-time"
627 |             DataType="&xml;time"/>
628 |         </Apply>
629 |         <AttributeValue
630 |             DataType="&xml;time">17h</AttributeValue>
631 |         </Apply>
632 |     </Apply>
633 | </Condition>
634 | </Rule>
635 |
636 | <!-- Manager role requirements rule -->
637 | <Rule RuleId="manager:role:requirements" Effect="Permit">
638 |     <Target>
639 |         <AnyOf>
640 |             <AllOf>
641 |                 <Match MatchId="&function;string-equal">
642 |                     <AttributeValue
643 |                         DataType="&xml;string">Steve</AttributeValue>
644 |                     <AttributeDesignator
645 |                         MustBePresent="false"
646 |                         Category="&subject-category;access-subject"
647 |                         AttributeId="&subject;subject-id"
648 |                         DataType="&xml;string"/>
649 |                     </Match>
650 |                 </AllOf>
651 |             </AnyOf>
652 |             <AnyOf>
653 |                 <AllOf>
654 |                     <Match MatchId="&function;anyURI-equal">
655 |                         <AttributeValue
656 |                             DataType="&xml;anyURI">&roles;:manager</AttributeValue>
657 |                         <AttributeDesignator
658 |                             MustBePresent="false"
659 |                             Category="&category;resource"
660 |                             AttributeId="&role;"
661 |                             DataType="&xml;anyURI"/>
662 |                         </Match>
663 |                     </AllOf>
664 |                 </AnyOf>
665 |             <AnyOf>
666 |                 <AllOf>
667 |                     <Match MatchId="&function;anyURI-equal">
668 |                         <AttributeValue
669 |                             DataType="&xml;anyURI">&actions;enableRole</AttributeValue>
670 |                         <AttributeDesignator
671 |                             MustBePresent="false"
672 |                             Category="&category;action"
673 |                             AttributeId="&action;action-id"
674 |                             DataType="&xml;anyURI"/>
675 |                         </Match>
676 |                     </AllOf>
677 |                 </AnyOf>
678 |             </Target>
679 |         </Rule>
680 | </Policy>

```

681 | *Listing – Role Assignment <Policy> Example*

682 4 Implementing the RBAC Model

683 {non-normative}

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

686 4.1 4.1 Core RBAC

687 {non-normative}

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

- 689 1. Users
- 690 2. **Roles**
- 691 3. Objects
- 692 4. Operations
- 693 5. **Permissions**

694 Users are implemented using XACML Subjects. Any of the XACML attribute `Category` values which are
695 semantically associated with subjects may be used, as appropriate.

696 **Roles** are expressed using one or more XACML Subject Attributes. The set of **roles** is very application-
697 and policy domain-specific, and it is very important that different uses of **roles** not be confused. For
698 these reasons, this profile does not attempt to define any standard set of **role** values, although this profile
699 does recommend use of a common `AttributeId` value of “urn:oasis:names:tc:xacml:2.0:subject:role”.
700 It is recommended that each application or policy domain agree on and publish a unique set of
701 `AttributeId` values, `DataType` values, and `<AttributeValue>` values that will be used for the
702 various **roles** relevant to that domain.

703 Objects are expressed using XACML Resources.

704 Operations are expressed using XACML Actions.

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

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

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

716 XACML allows multiple **role** attributes or **role** attribute values to be associated with a given Subject. If a
717 Subject has multiple **roles** enabled, then any Role `<PolicySet>` instance applying to any of those **roles**
718 may be evaluated, and the **permissions** in the corresponding Permission `<PolicySet>` will be
719 permitted. As described in Section 1.10: Multi-Role Permissions, it is even possible to define policies that
720 require a given Subject to have multiple **role** attributes or values enabled at the same time. In this case,
721 the **permissions** associated with the multiple-**role** requirement will apply only to a Subject having all the
722 necessary **role** attributes and values at the time an XACML Request Context is presented to the PDP for
723 evaluation.

724 The Permission `<PolicySet>` associated with a given **role** may allow access to multiple resources
725 using multiple actions. XACML has a rich set of constructs for composing **permissions**, so there are
726 multiple ways in which multi-permission **roles** may be expressed. Any Role A may be associated with a

727 Permission <PolicySet> B by including a <PolicySetIdReference> to Permission <PolicySet>
728 B in the Permission <PolicySet> associated with the Role A. In this way, the same set of **permissions**
729 may be associated with more than one **role**.

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

734 4.2 1.2 — Hierarchical RBAC

735 {non-normative}

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

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

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

749 5 Profile

750 5.1 Roles and Role Attributes

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

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

759 5.2 Role Assignment or Enablement

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

769 5.3 Access Control

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

773 For each **role**, one Role `<PolicySet>` SHALL be defined. Such a `<PolicySet>` SHALL contain a
774 `<Target>` element that makes the `<PolicySet>` applicable only to Subjects having the XACML
775 Attribute associated with the given **role**; the `<Target>` element SHALL NOT restrict the Resource,
776 Action, or Environment. Each Role `<PolicySet>` SHALL contain a single `<PolicySetIdReference>`
777 element that references the unique Permission `<PolicySet>` associated with the **role**. The Role
778 `<PolicySet>` SHALL NOT contain any other `<Policy>`, `<PolicySet>`, `<PolicyIdReference>`, or
779 `<PolicySetIdReference>` elements.

780 For each **role**, one Permission `<PolicySet>` SHALL be defined. Such a `<PolicySet>` SHALL contain
781 `<PolicySet>`, `<Policy>` and `<Rule>` elements that specify the types of access permitted to Subjects
782 having the given **role**. The `<Target>` of the `<PolicySet>` and its included or referenced
783 `<PolicySet>`, `<Policy>`, and `<Rule>` elements SHALL NOT limit the Subjects to which the
784 Permission `<PolicySet>` is applicable.

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

789 A Permission `<PolicySet>` MAY include a `HasPrivilegesOfRole <Policy>`. Such a `<Policy>` SHALL
790 have a `<Rule>` element with an effect of “Permit”. This Rule SHALL permit any Subject to perform an
791 Action with an Attribute having an `AttributeId` of `&action;action-id`, a `DataType` of `&xml:anyURI`, and
792 an `<AttributeValue>` having a value of `&actions;hasPrivilegesOfRole` on a Resource having an

793 Attribute that is the **role** to which the Permission <PolicySet> applies (for example, an AttributeId
794 of &role;, a DataType of &xml:anyURI, and an <AttributeValue> whose value is the URI of the
795 specific **role** value). Note that the **role** Attribute, which is a Subject Attribute in a Role <PolicySet>
796 <Target>, is treated as a Resource Attribute in a HasPrivilegesOfRole <Policy>.

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

799 6 Identifiers

800 This profile defines the following URN identifiers.

801 6.1 Profile Identifier

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

804 urn:oasis:names:tc:xacml:3.0:profiles:rbac:core-hierarchical

805 6.2 Role Attribute

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

807 urn:oasis:names:tc:xacml:2.0:subject:role

808 ~~6.3 SubjectCategory~~

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

811 ~~urn:oasis:names:tc:xacml:2.0:subject-category:role-enablement-authority~~

812 ~~6.46.3 Action Attribute Values~~

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

815 urn:oasis:names:tc:xacml:2.0:actions:hasPrivilegesOfRole

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

818 ~~urn:oasis:names:tc:xacml:2.0:actions:enableRole~~

819 **7 Conformance**

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

821 **7.1 As a policy processor**

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

824 **7.2 As an XACML request generator**

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

827

Appendix A. Acknowledgements

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

830

831 Anil Saldhana

832 Anil Tappetla

833 Anne Anderson

834 Anthony Nadalin

835 Bill Parducci

836 Craig Forster

837 David Chadwick

838 David Staggs

839 Dilli Arumugam

840 Duane DeCouteau

841 Erik Rissanen

842 Gareth Richards

843 Hal Lockhart

844 Jan Herrmann

845 John Tolbert

846 Ludwig Seitz

847 Michiharu Kudo

848 Naomaru Itoi

849 Paul Tyson

850 Prateek Mishra

851 Rich Levinson

852 Ronald Jacobson

853 Seth Proctor

854 Sridhar Muppidi

855 Tim Moses

856 Vernon Murdoch

Appendix B. Revision History

Revision	Date	Editor	Changes Made
WD 1	[Rev Date]	Erik Rissanen	Initial update to XACML 3.0.
WD 2	28 Dec 2007	Erik Rissanen	Update to the current OASIS template.
WD 3	4 Nov 2008	Erik Rissanen	Fixed typos in the examples.
WD 4	5 Apr 2009	Erik Rissanen	Editorial cleanups. Added conformance section.
WD 5	14 Dec 2009	Erik Rissanen	Also allow <PolicySet> in permission policysset.
WD 06	17 Dec 2009	Erik Rissanen	Fixed formatting issues Updated acknowledgments
WD 07	12 Jan 2010	Erik Rissanen	Updated cross references. Corrected examples so they are valid against the XACML schema. Updated acknowledgments
WD 08	8 Mar 2010	Erik Rissanen	Updated cross references Fixed OASIS formatting issues Removed reference to XACML 2.0 intro
WD 09	24 May 2011	Erik Rissanen	Also allow <PolicySet> in permission policysset in the non-normative text in section 1.8.
WD 10	23 Jan 2014	Erik Rissanen	Migrated to current OASIS document template.
WD 11	15 May 2014	Erik Rissanen	Removed examples of XACML based role enablement authorities.