



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

Committee Specification 01

10 August 2010

Specification URIs:

This Version:

<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-cs-01-en.html>
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-cs-01-en.doc> (Authoritative)
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-cs-01-en.pdf>

Previous Version:

<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-cd-03-en.html>
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-cd-03-en.doc> (Authoritative)
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-cd-03-en.pdf>

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)
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-en.pdf>

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, CD 03](#)

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® 2010. 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.....	12
2.5	HasPrivilegesOfRole Policies and Requests	12
3	Assigning and Enabling Role Attributes.....	15
4	Implementing the RBAC Model	18
4.1	1.1 Core RBAC.....	18
4.2	1.2 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	SubjectCategory	22
6.4	Action Attribute Values	22
7	Conformance.....	23
7.1	As a policy processor.....	23
7.2	As an XACML request generator	23
A.	Acknowledgements	24
B.	Revision History	25

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 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 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].

43 Senior role

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

46 1.2 XML Entity Declarations

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

49

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

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

64 1.3 Terminology

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

68 1.4 Normative References

- 69 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
70 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 71 [XACML] OASIS Committee Specification 01, eXtensible access control markup language
72 (XACML) Version 3.0. August 2010. [http://docs.oasis-open.org/xacml/3.0/xacml-
73 3.0-core-spec-cs-01-en.doc](http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.doc)

74 1.5 Non-Normative References

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

84 1.6 Scope

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

88 The policies specified in this profile can answer three types of questions:

- 89 1. If a subject has **roles** R_1, R_2, \dots, R_n enabled, can subject X access a given resource using a given
90 action?
- 91 2. Is subject X allowed to have **role** R_i enabled?
- 92 3. If a subject has **roles** R_1, R_2, \dots, R_n enabled, does that mean the subject will have **permissions**
93 associated with a given **role** R_i ? That is, is **role** R_i either equal to or junior to any of **roles** $R_1, R_2,$
94 \dots, R_n ?

95 The policies specified in this profile do not answer the question “What set of **roles** does subject X have?”
96 That question must be handled by a **Role Enablement Authority**, and not directly by an XACML PDP.
97 Such an entity may make use of XACML policies, but will need additional information. See Section 3:
98 Assigning and Enabling Role Attributes for more information about **Role Enablement Authorities**.

99 The policies specified in this profile assume all the **roles** for a given subject have already been enabled at
100 the time an authorization decision is requested. They do not deal with an environment in which **roles**
101 must be enabled dynamically based on the resource or actions a subject is attempting to perform. For
102 this reason, the policies specified in this profile also do not deal with static or dynamic “Separation of
103 Duty” (see [ANSI-RBAC]). A future profile may address the requirements of this type of environment.

104 1.7 Role

105 In this profile, **roles** are expressed as XACML Subject Attributes. There are two exceptions: in a Role
106 Assignment `<PolicySet>` or `<Policy>` and in a HasPrivilegesOfRole `<Policy>`, the **role** appears as
107 a Resource Attribute. See Section 2.5: HasPrivilegesOfRole Policies and Requests and Section 3:
108 Assigning and Enabling Role Attributes for more information.

109 **Role** attributes may be expressed in either of two ways, depending on the requirements of the application
110 environment. In some environments there may be a small number of “**role** attributes”, where the name of
111 each such attribute is some name indicating “**role**”, and where the value of each such attribute indicates
112 the name of the **role** held. For example, in this first type of environment, there may be one “**role** attribute”
113 having the `AttributeId` “&role;” (this profile recommends use of this identifier). The possible **roles** are
114 values for this one attribute, and might be “&roles;officer”, “&roles;manager”, and “&roles;employee”. This
115 way of expressing **roles** works best with the XACML way of expressing policies. This method of
116 identifying **roles** is also most conducive to interoperability.

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

123 XACML supports multiple subjects per access request, indicating various entities that may be involved in
124 making the request. For example, there is usually a human user who initiates the request, at least
125 indirectly. There are usually one or more applications or code bases that generate the actual low-level
126 access request on behalf of the user. There is some computing device on which the application or code
127 base is executing, and this device may have an identity such an IP address. XACML identifies each such
128 Subject with a `Category` xml attribute in the `<Attributes>` element that indicates the type of subject
129 being described. For example, the human user has a `Category` of `&subject-category;access-subject;`
130 the application that generates the access request has a `Category` of `&subject-category;codebase` and
131 so on. In this profile, a **role** attribute may be associated with any of the categories of subjects involved in
132 making an access request.

133 1.8 Policies

134 In this profile, four types of policies are specified.

- 135 1. **Role <PolicySet> or RPS**: a `<PolicySet>` that associates holders of a given **role** attribute
136 and value with a `Permission <PolicySet>` that contains the actual **permissions** associated with
137 the given **role**. The `<Target>` element of a `Role <PolicySet>` limits the applicability of the
138 `<PolicySet>` to subjects holding the associated **role** attribute and value. Each `Role`

- 139 <PolicySet> references a single corresponding Permission <PolicySet> but does not
140 contain or reference any other <Policy> or <PolicySet> elements.
- 141 2. **Permission <PolicySet> or PPS:** a <PolicySet> that contains the actual **permissions**
142 associated with a given **role**. It contains <Policy> elements and <Rules> that describe the
143 resources and actions that subjects are permitted to access, along with any further conditions on
144 that access, such as time of day. A given Permission <PolicySet> may also contain
145 references to Permission <PolicySet>s associated with other **roles** that are junior to the given
146 **role**, thereby allowing the given Permission <PolicySet> to inherit all **permissions** associated
147 with the **role** of the referenced Permission <PolicySet>. The <Target> element of a
148 Permission <PolicySet>, if present, must not limit the subjects to which the <PolicySet> is
149 applicable.
 - 150 3. **Role Assignment <Policy> or <PolicySet>:** a <Policy> or <PolicySet> that defines
151 which **roles** can be enabled or assigned to which subjects. It may also specify restrictions on
152 combinations of **roles** or total number of **roles** assigned to or enabled for a given subject. This
153 type of policy is used by a **Role Enablement Authority**. Use of a Role Assignment <Policy> or
154 <PolicySet> is optional.
 - 155 4. **HasPrivilegesOfRole <Policy>:** a <Policy> in a Permission <PolicySet> that supports
156 requests asking whether a subject has the privileges associated with a given **role**. If this type of
157 request is to be supported, then a HasPrivilegesOfRole <Policy> must be included in each
158 Permission <PolicySet>. Support for this type of <Policy>, and thus for requests asking
159 whether a subject has the privileges associated with a given **role**, is optional.

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

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

173 1.9 Multi-Role Permissions

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

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

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

187 2 Example

188 {non-normative}

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

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

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

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

208 The actual XACML policies implementing this example follow. An example of a Role Assignment Policy is
209 included in Section 3: Assigning and Enabling Role Attributes.

210 2.1 Permission <PolicySet> for the manager role

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

214

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

```

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

```

265 *Listing 1 Permission <PolicySet> for managers*

266 2.2 Permission <PolicySet> for employee role

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

```

271
272 <PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
273     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
274     xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-
275 core-v3-schema-wd-17.xsd"
276     PolicySetId="PPS:employee:role"
277     Version="1.0"
278     PolicyCombiningAlgId="&policy-combine;permit-overrides">
279
280     <Target/>
281     <!-- Permissions specifically for the employee role -->
282     <Policy PolicyId="Permissions:specifically:for:the:employee:role"
283         Version="1.0"
284         RuleCombiningAlgId="&rule-combine;permit-overrides">
285         <Target/>
286         <!-- Permission to create a purchase order -->
287         <Rule RuleId="Permission:to:create:a:purchase:order" Effect="Permit">
288             <Target>
289                 <AnyOf>
290                     <AllOf>
291                         <Match MatchId="&function;string-equal">
292                             <AttributeValue
293                                 DataType="&xml;string">purchase order</AttributeValue>
294                             <AttributeDesignator
295                                 MustBePresent="false"
296                                 Category="&category;resource"
297                                 AttributeId="&resource;resource-id"

```

```

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

```

319 *Listing 2 Permission <PolicySet> for employees*

320 2.3 Role <PolicySet> for the manager role

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

```

325
326 <PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
327     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
328     xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-
329 core-v3-schema-wd-17.xsd"
330     PolicySetId="RPS:manager:role"
331     Version="1.0"
332     PolicyCombiningAlgId="&policy-combine;permit-overrides">
333     <Target>
334         <AnyOf>
335             <AllOf>
336                 <Match MatchId="&function:anyURI-equal">
337                     <AttributeValue
338                         DataType="&xml:anyURI">&roles;manager</AttributeValue>
339                     <AttributeDesignator
340                         MustBePresent="false"
341                         Category="&subject-category;access-subject"
342                         AttributeId="&role;"
343                         DataType="&xml:anyURI"/>
344                 </Match>
345             </AllOf>
346         </AnyOf>
347     </Target>
348
349     <!-- Use permissions associated with the manager role -->
350     <PolicySetIdReference>PPS:manager:role</PolicySetIdReference>
351 </PolicySet>

```

352 *Listing 3 Role <PolicySet> for managers*

353 2.4 Role <PolicySet> for employee role

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

358

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

385 *Listing 4 Role <PolicySet> for employees*

386 2.5 HasPrivilegesOfRole Policies and Requests

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

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

392

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

```

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

```

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

430

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

433

```

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

```

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

471

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

474

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

501 Listing 7 Example of *HasPrivilegesOfRole* Request

502

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

3 Assigning and Enabling Role Attributes

507

508 {non-normative}

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

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

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

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

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

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

542

```
543 <Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"  
544   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
545   xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-  
546   core-v3-schema-wd-17.xsd"  
547   PolicyId="Role:Assignment:Policy"  
548   Version="1.0"  
549   RuleCombiningAlgId="&rule-combine;permit-overrides">  
550  
551   <Target/>  
552   <!-- Employee role requirements rule -->  
553   <Rule RuleId="employee:role:requirements" Effect="Permit">  
554     <Target>  
555       <AnyOf>  
556         <AllOf>  
557           <Match MatchId="&function:string-equal">  
558             <AttributeValue
```

```

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

```

```

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

```

679 *Listing 8 Role Assignment <Policy> Example*

680 4 Implementing the RBAC Model

681 {non-normative}

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

684 4.1 1.1 Core RBAC

685 {non-normative}

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

- 687 1. Users
- 688 2. Roles
- 689 3. Objects
- 690 4. Operations
- 691 5. Permissions

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

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

701 **Objects** are expressed using XACML Resources.

702 **Operations** are expressed using XACML Actions.

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

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

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

714 XACML allows multiple **role** attributes or **role** attribute values to be associated with a given Subject. If a
715 Subject has multiple **roles** enabled, then any Role `<PolicySet>` instance applying to any of those **roles**
716 may be evaluated, and the **permissions** in the corresponding Permission `<PolicySet>` will be
717 permitted. As described in Section 1.9: Multi-Role Permissions, it is even possible to define policies that
718 require a given Subject to have multiple **role** attributes or values enabled at the same time. In this case,
719 the **permissions** associated with the multiple-**role** requirement will apply only to a Subject having all the
720 necessary **role** attributes and values at the time an XACML Request Context is presented to the PDP for
721 evaluation.

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

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

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

732 4.2 1.2 Hierarchical RBAC

733 {non-normative}

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

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

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

747 5 Profile

748 5.1 Roles and Role Attributes

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

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

757 5.2 Role Assignment or Enablement

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

767 5.3 Access Control

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

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

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

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

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

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

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

797 6 Identifiers

798 This profile defines the following URN identifiers.

799 6.1 Profile Identifier

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

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

803 6.2 Role Attribute

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

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

806 6.3 SubjectCategory

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

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

810 6.4 Action Attribute Values

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

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

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

816 urn:oasis:names:tc:xacml:2.0:actions:enableRole

817 **7 Conformance**

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

819 **7.1 As a policy processor**

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

822 **7.2 As an XACML request generator**

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

825 **A. Acknowledgements**

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

828

829 Anil Saldhana

830 Anil Tappetla

831 Anne Anderson

832 Anthony Nadalin

833 Bill Parducci

834 Craig Forster

835 David Chadwick

836 David Staggs

837 Dilli Arumugam

838 Duane DeCouteau

839 Erik Rissanen

840 Gareth Richards

841 Hal Lockhart

842 Jan Herrmann

843 John Tolbert

844 Ludwig Seitz

845 Michiharu Kudo

846 Naomaru Itoi

847 Paul Tyson

848 Prateek Mishra

849 Rich Levinson

850 Ronald Jacobson

851 Seth Proctor

852 Sridhar Muppidi

853 Tim Moses

854 Vernon Murdoch

855

856

857

B. Revision History

858 [optional; should not be included in OASIS Standards]

859

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 policyset.
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

860