



XACML Profile for Role Based Access Control (RBAC)

Committee Specification 01, 13 February 2004

Document identifier:

cs-xacml-rbac-profile-01

Location:

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

Editor:

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

Abstract:

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

Status:

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

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

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

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

26 **Table of Contents**

27 1 Introduction (non-normative).....3
28 1.1 Notation.....3
29 1.2 Terminology.....3
30 1.3 Role.....4
31 1.4 Policies.....4
32 1.5 Multi-Role Permissions.....5
33 2 Example (non-normative).....6
34 2.1 Permission <PolicySet> for the manager role.....6
35 2.2 Permission <PolicySet> for employee role.....7
36 2.3 Role <PolicySet> for the manager role.....8
37 2.4 Role <PolicySet> for employee role.....8
38 3 Assigning and Enabling Role Attributes (non-normative)..... 9
39 4 Implementing the RBAC Model (non-normative).....12
40 4.1 Core RBAC.....12
41 4.2 Hierarchical RBAC.....13
42 4.3 Separation of Duty.....13
43 5 Profile (normative).....16
44 5.1 Role Assignment or Enablement.....16
45 5.2 Access Control.....16
46 6 References.....17
47 6.1 Normative References.....17
48 6.2 Non-normative References.....17

1 Introduction (non-normative)

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

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

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

1.1 Notation

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

```
<!-- ENTITY xacml -->
<!-- ENTITY xml -->
<!-- ENTITY rule-combine -->
<!-- ENTITY policy-combine -->
<!-- ENTITY function -->
<!-- ENTITY subject-category -->
<!-- ENTITY subject -->
<!-- ENTITY resource -->
<!-- ENTITY action -->
<!-- ENTITY environment -->
```

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

1.2 Terminology

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

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

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

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

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

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

97 **PPS** – Permission `<PolicySet>`. See *Section 1.4 Policies*.
98 **RBAC** – Role based access control. A model for controlling access to resources where permitted
99 actions on resources are identified with roles rather than with individual subject identities.
100 **RPS** – Role `<PolicySet>`. See *Section 1.4 Policies*.
101 **role** – A job function within the context of an organization that has associated semantics regarding the
102 authority and responsibility conferred on the user assigned to the role [RBAC].
103 **senior role** – In a role hierarchy, Role A is *senior* to Role B if Role A inherits all the permissions
104 associated with Role B.
105 **policy** – A set of rules indicating which subjects are permitted to access which resources using which
106 actions under which conditions.

107 **1.3 Role**

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

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

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`,
121 `urn:someapp:attributes:manager-role`, and `urn:someapp:attributes:employee-role`.
122 In this case the value of the attribute may be empty or it may contain various parameters associated with
123 the role. XACML policies can handle roles expressed in this 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 request on behalf of the user. There is some computing device on which the application or code base is
128 executing, and this device may have an identity such an IP address. XACML identifies each such
129 Subject with a `SubjectCategory` xml attribute that indicates the type of subject being described. For
130 example, the human user has a `SubjectCategory` of `&subject-category;access-subject;`
131 (this is the default category); the application that generates the access request has a
132 `SubjectCategory` of `&subject-category;codebase;` and so on. In this Profile, a role
133 attribute may be associated with any of the categories of subjects involved in making an access request.

134 **1.4 Policies**

135 In this Profile, there are four types of policies.

- 136 1. **Role `<PolicySet>` or RPS** : a `<PolicySet>` that associates holders of a given role attribute with a
137 Permission `<PolicySet>` that contains the actual permissions associated with the given role. The
138 `<Target>` element of a Role `<PolicySet>` limits the applicability of the `<PolicySet>` to subjects
139 holding the given role attribute. Each Role `<PolicySet>` references a single corresponding
140 Permission `<PolicySet>` but does not contain any other `<Policy>` or `<PolicySet>` elements.
- 141 2. **Permission `<PolicySet>` or PPS**: a `<PolicySet>` that contains the actual permissions associated
142 with a given role. It contains `<Policy>` elements and `<Rules>` that describe the resources and
143 actions that subjects are permitted to access, along with any further conditions on that access, such
144 as time of day. A given Permission `<PolicySet>` may also contain references to Permission

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

149 **3. Separation of Duty <PolicySet>**: a <PolicySet> that defines restrictions on the set of roles that
150 can be exercised by a given Subject. Such a <PolicySet> contains <Policy> and <Rule>
151 elements that specify the role set restrictions. The Separation of Duty <PolicySet> also contains
152 references to all the Role <PolicySet> instances that are subject to Separation of Duty restrictions.
153 Use of a Separation of Duty <PolicySet> is optional.

154 **4. Role Assignment <Policy> or <PolicySet>**: a <Policy> or <PolicySet> that defines which
155 roles can be enabled or assigned to which subjects. It may also specify restrictions on combinations
156 of roles or total number of roles assigned to or enabled for a given subject. This type of policy is used
157 by the entity that assigns role attributes to users or by the entity that enables role attributes during a
158 user's session. Use of a Role Assignment <Policy> or <PolicySet> is optional.

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

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

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

176 **1.5 Multi-Role Permissions**

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

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

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

2 Example (non-normative)

191

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

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

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

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

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

2.1 Permission <PolicySet> for the *manager* role

214

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

```
1. <PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy"  
2.   PolicySetId="PPS:manager:role"  
3.   PolicyCombiningAlgId="&policy-combine;permit-overrides">  
4.   <Target>  
5.     <Subjects><AnySubject/></Subjects>  
6.     <Resources><AnyResource/></Resources>  
7.     <Actions><AnyAction/></Actions>  
8.   </Target>  
9.  
10.   <!-- Permissions specifically for the manager role -->  
11.   <Policy PolicyId="Permissions:specifically:for:the:manager:role"  
12.     RuleCombiningAlgId="&rule-combine;permit-overrides">  
13.     <Target>  
14.       <Subjects><AnySubject/></Subjects>  
15.       <Resources><AnyResource/></Resources>  
16.       <Actions><AnyAction/></Actions>  
17.     </Target>  
18.  
19.     <!-- Permission to sign a purchase order -->  
20.     <Rule RuleId="Permission:to:sign:a:purchase:order"  
21.       Effect="Permit">  
22.       <Target>  
23.         <Subjects><AnySubject/></Subjects>  
24.         <Resources>  
25.           <Resource>  
26.             <ResourceMatch MatchId="&function:string-match">  
27.               <AttributeValue  
28.                 DataType="&xml:string">purchase order</AttributeValue>  
29.               <ResourceAttributeDesignator  
30.                 AttributeId="&resource;resource-id"  
31.                 DataType="&xml:string"/>  
           </ResourceMatch>
```

```

32.         </Resource>
33.     </Resources>
34.     <Actions>
35.         <Action>
36.             <ActionMatch MatchId="&function;string-match">
37.                 <AttributeValue
38.                     DataType="&xml;string">sign</AttributeValue>
39.                 <ActionAttributeDesignator
40.                     AttributeId="&action;action-id"
41.                     DataType="&xml;string"/>
42.             </ActionMatch>
43.         </Action>
44.     </Actions>
45. </Target>
46. </Rule>
47. </Policy>
48.
49.     <!-- Include permissions associated with employee role -->
50.     <PolicySetIdReference>PPS:employee:role</PolicySetIdReference>
51. </PolicySet>

```

Table 1 Permission <PolicySet> for managers

2.2 Permission <PolicySet> for employee role

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

```

52. <PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy"
53.     PolicySetId="PPS:employee:role"
54.     PolicyCombiningAlgId="&policy-combine;permit-overrides">
55.     <Target>
56.         <Subjects><AnySubject/></Subjects>
57.         <Resources><AnyResource/></Resources>
58.         <Actions><AnyAction/></Actions>
59.     </Target>
60.
61.     <!-- Permissions specifically for the employee role -->
62.     <Policy PolicyId="Permissions:specifically:for:the:employee:role"
63.         RuleCombiningAlgId="&rule-combine;permit-overrides">
64.         <Target>
65.             <Subjects><AnySubject/></Subjects>
66.             <Resources><AnyResource/></Resources>
67.             <Actions><AnyAction/></Actions>
68.         </Target>
69.
70.         <!-- Permission to create a purchase order -->
71.         <Rule RuleId="Permission:to:create:a:purchase:order"
72.             Effect="Permit">
73.             <Target>
74.                 <Subjects><AnySubject/></Subjects>
75.                 <Resources>
76.                     <Resource>
77.                         <ResourceMatch MatchId="&function;string-match">
78.                             <AttributeValue
79.                                 DataType="&xml;string">purchase order</AttributeValue>
80.                             <ResourceAttributeDesignator
81.                                 AttributeId="&resource;resource-id"
82.                                 DataType="&xml;string"/>
83.                         </ResourceMatch>
84.                     </Resource>
85.                 </Resources>
86.                 <Actions>
87.                     <Action>
88.                         <ActionMatch MatchId="&function;string-match">
89.                             <AttributeValue
90.                                 DataType="&xml;string">create</AttributeValue>
91.                             <ActionAttributeDesignator

```

```

92.         DataType="&xml;string"/>
93.     </ActionMatch>
94. </Action>
95. </Actions>
96. </Target>
97. </Rule>
98. </Policy>
99. </PolicySet>

```

Table 2 Permission <PolicySet> for employees

222 2.3 Role <PolicySet> for the manager role

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

```

100. <PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy"
101.     PolicySetId="RPS:manager:role"
102.     PolicyCombiningAlgId="&policy-combine;permit-overrides">
103. <Target>
104.     <Subjects>
105.         <Subject>
106.             <SubjectMatch MatchId="&function;string-equal">
107.                 <AttributeValue
108.                     DataType="&xml;string">manager</AttributeValue>
109.                 <SubjectAttributeDesignator
110.                     AttributeId="urn:someapp:attributes:role"
111.                     DataType="&xml;string"/>
112.             </SubjectMatch>
113.         </Subject>
114.     </Subjects>
115.     <Resources><AnyResource/></Resources>
116.     <Actions><AnyAction/></Actions>
117. </Target>
118.
119. <!-- Use permissions associated with the manager role -->
120. <PolicySetIdReference>PPS:manager:role</PolicySetIdReference>
121. </PolicySet>

```

Table 3 Role <PolicySet> for managers

227 2.4 Role <PolicySet> for employee role

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


```

120.     <PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy"
121.         PolicySetId="RPS:employee:role"
122.         PolicyCombiningAlgId="&policy-combine;permit-overrides">
123.     <Target>
124.         <Subjects>
125.             <Subject>
126.                 <SubjectMatch MatchId="&function;string-equal">
127.                     <AttributeValue
128.                         DataType="&xml:string">employee</AttributeValue>
129.                     <SubjectAttributeDesignator
130.                         AttributeId="urn:someapp:attributes:role"
131.                         DataType="&xml:string"/>
132.                 </SubjectMatch>
133.             </Subject>
134.         </Subjects>
135.         <Resources><AnyResource/></Resources>
136.         <Actions><AnyAction/></Actions>
137.     </Target>
138.
139.     <!-- Use permissions associated with the employee role -->
140.     <PolicySetIdReference>PPS:employee:role</PolicySetIdReference>
141. </PolicySet>

```

Table 4 Role <PolicySet> for employees

3 Assigning and Enabling Role Attributes (non-normative)

232

233

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

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

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

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

247

```
147.     <!-- Employee role requirements rule -->  
148.     <Rule RuleId="employee:role:requirements" Effect="Permit">  
149.         <Target>  
150.             <Subjects>  
151.                 <Subject>  
152.                     <SubjectMatch MatchId="&function;string-equal">  
153.                         <AttributeValue  
154.                             DataType="&xml:string">Seth</AttributeValue>  
155.                         <SubjectAttributeDesignator  
156.                             AttributeId="&subject;subject-id"  
157.                             DataType="&xml:string"/>  
158.                     </SubjectMatch>  
159.                 </Subject>  
160.                 <Subject>  
161.                     <SubjectMatch MatchId="&function;string-equal">  
162.                         <AttributeValue  
163.                             DataType="&xml:string">Anne</AttributeValue>  
164.                         <SubjectAttributeDesignator  
165.                             AttributeId="&subject;subject-id"  
166.                             DataType="&xml:string"/>  
167.                     </SubjectMatch>  
168.                 </Subject>  
169.             </Subjects>  
170.             <Resources>  
171.                 <Resource>  
172.                     <ResourceMatch MatchId="&function;string-equal">  
173.                         <AttributeValue  
174.                             DataType="&xml:string">employee</AttributeValue>
```

```

175.         <ResourceAttributeDesignator
176.             AttributeId="urn:someapp:attributes:role"
177.             DataType="&xml:string"/>
178.         </ResourceMatch>
179.     </Resource>
180. </Resources>
181. <Actions>
182.     <Action>
183.         <ActionMatch MatchId="&function;string-equal">
184.             <AttributeValue
185.                 DataType="&xml:string">enable</AttributeValue>
186.             <ActionAttributeDesignator
187.                 AttributeId="&action;action-id"
188.                 DataType="&xml:string"/>
189.         </ActionMatch>
190.     </Action>
191. </Actions>
192. </Target>
193. <Condition FunctionId="&function;and">
194.     <Apply FunctionId="&function;time-greater-than-or-equal">
195.         <Apply FunctionId="&function;time-one-and-only">
196.             <EnvironmentAttributeDesignator
197.                 AttributeId="&environment;current-time"
198.                 DataType="&xml;time"/>
199.         </Apply>
200.         <AttributeValue
201.             DataType="&xml;time">9h</AttributeValue>
202.     </Apply>
203.     <Apply FunctionId="&function;time-less-than-or-equal">
204.         <Apply FunctionId="&function;time-one-and-only">
205.             <EnvironmentAttributeDesignator
206.                 AttributeId="&environment;current-time"
207.                 DataType="&xml;time"/>
208.         </Apply>
209.         <AttributeValue
210.             DataType="&xml;time">17h</AttributeValue>
211.     </Apply>
212. </Condition>
213. </Rule>

```

248

```

214. <!-- Manager role requirements rule -->
215. <Rule RuleId="manager:role:requirements" Effect="Permit">
216.     <Target>
217.         <Subjects>
218.             <Subject>
219.                 <SubjectMatch MatchId="&function;string-equal">
220.                     <AttributeValue
221.                         DataType="&xml:string">Steve</AttributeValue>
222.                     <SubjectAttributeDesignator
223.                         AttributeId="&subject;subject-id"
224.                         DataType="&xml:string"/>
225.                 </SubjectMatch>
226.             </Subject>
227.         </Subjects>
228.     </Resources>

```

```

229.     <Resource>
230.         <ResourceMatch MatchId="&function;string-equal">
231.             <AttributeValue
232.                 DataType="&xml;string">manager</AttributeValue>
233.             <ResourceAttributeDesignator
234.                 AttributeId="urn:someapp:attributes:role"
235.                 DataType="&xml;string"/>
236.         </ResourceMatch>
237.     </Resource>
238. </Resources>
239. <Actions>
240.     <Action>
241.         <ActionMatch MatchId="&function;string-equal">
242.             <AttributeValue
243.                 DataType="&xml;string">enable</AttributeValue>
244.             <ActionAttributeDesignator
245.                 AttributeId="&action;action-id"
246.                 DataType="&xml;string"/>
247.         </ActionMatch>
248.     </Action>
249. </Actions>
250. </Target>
251. </Rule>
252. </Policy>

```

Table 5 Role Assignment <Policy> Example

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

251 4 Implementing the RBAC Model (non-normative)

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

254 4.1 Core RBAC

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

256 1. Users

257 2. Roles

258 3. Objects

259 4. Operations

260 5. Permissions

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

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

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

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

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

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

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

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

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

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

299 4.2 Hierarchical RBAC

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

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

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

313 4.3 Separation of Duty

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

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

323 Separation of Duty `<PolicySet>`

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

```
253.     <PolicySet xmlns="urn:oasis:names:tc:xacml:1.0:policy"  
254.         PolicySetId="Separation:of:Duty:PolicySet"  
255.         PolicyCombiningAlgId="&policy-combine;deny-overrides">  
256.         <Target>  
257.             <Subjects><AnySubject/></Subjects>  
258.             <Resources><AnyResource/></Resources>  
259.             <Actions><AnyAction/></Actions>  
260.         </Target>  
261.         <!-- Disallow simultaneous contractor and employee roles -->  
262.         <Policy PolicyId="contractor:AND:employee:disallowed"  
263.             RuleCombiningAlgId="&rule-combine;deny-overrides">  
264.             <Target>  
265.                 <Subjects>  
266.                     <Subject>  
267.                         <SubjectMatch MatchId="&function;string-equal">  
268.                             <AttributeValue  
269.                                 DataType="&xml:string">employee</AttributeValue>
```

```

270.         <SubjectAttributeDesignator
271.             AttributeId="urn:someapp:attributes:role"
272.             DataType="&xml:string"/>
273.         </SubjectMatch>
274.         <SubjectMatch MatchId="&function:string-equal">
275.             <AttributeValue
276.                 DataType="&xml:string">contractor</AttributeValue>
277.             <SubjectAttributeDesignator
278.                 AttributeId="urn:someapp:attributes:role"
279.                 DataType="&xml:string"/>
280.             </SubjectMatch>
281.         </Subject>
282.     </Subjects>
283.     <Resources><AnyResource/></Resources>
284.     <Actions><AnyAction/></Actions>
285. </Target>
286. <Rule RuleId="Deny:target:role:combination" Effect="Deny"/>
287. </Policy>
288.
289. <!-- Reference the Role PolicySets that are subject
290.      to separation of duty -->
291. <PolicySetIdReference>RPS:employee:role</PolicySetIdReference>
292. <PolicySetIdReference>RPS:contractor:role</PolicySetIdReference>
293. <PolicySetIdReference>RPS:manager:role</PolicySetIdReference>
</PolicySet>

```

Table 6 Separation of Duty <PolicySet> Example

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

332 **Role Assignment <PolicySet>**

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

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

```

294.     <Rule RuleId="Permission:to:hold:employee:role" Effect="Permit">
295.         <Target>
296.             <Subjects>
297.                 <Subject>
298.                     <SubjectMatch MatchId="&function:string-equal">
299.                         <AttributeValue
300.                             DataType="&xml:string">Seth</AttributeValue>
301.                         <SubjectAttributeDesignator
302.                             AttributeId="&subject;subject-id"
303.                             DataType="&xml:string"/>
304.                         </SubjectMatch>
305.                     </Subject>
306.                 <Subject>
307.                     <SubjectMatch MatchId="&function:string-equal">
308.                         <AttributeValue
309.                             DataType="&xml:string">Anne</AttributeValue>
310.                         <SubjectAttributeDesignator
311.                             AttributeId="&subject;subject-id"
312.                             DataType="&xml:string"/>
313.                         </SubjectMatch>
314.                     </Subject>
315.                 </Subjects>
316.             <Resources><AnyResource/></Resources>
317.             <Actions>
318.                 <Action>
319.                     <ActionMatch MatchId="&function:string-equal">

```

```

319.         <AttributeValue
320.             DataType="&xml:string">enable</AttributeValue>
321.         <ActionAttributeDesignator
322.             AttributeId="&action;action-id"
323.             DataType="&xml:string"/>
324.     </ActionMatch>
325. </Action>
326. </Actions>
327. </Target>
328. <Condition FunctionId="&function;integer-less-than-or-equal">
329.     <Apply FunctionId="&function;string-bag-size">
330.         <ResourceAttributeDesignator
331.             AttributeId="urn:someapp:attributes:role"
332.             DataType="&xml:string"/>
333.     </Apply>
334.     <AttributeValue
335.         DataType="&xml:string">2</AttributeValue>
336. </Condition>
</Rule>

```

Table 7 Separation of Duty <Rule> Example

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

346 5 Profile (normative)

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

352 5.1 Role Assignment or Enablement

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

356 5.2 Access Control

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

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

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

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

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

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

383 **6 References**

384 **6.1 Normative References**

- 385 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, IETF
386 RFC 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>
387 **[XACML]** T. Moses, ed., *OASIS eXtensible Access Control Markup Language (XACML)*
388 *Version 1.1*, [http://www.oasis-open.org/committees/xacml/repository/cs-xacml-](http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf)
389 [specification-1.1.pdf](http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf), Committee Specification, 24 July 2003.

390 **6.2 Non-normative References**

- 391 **[RBAC]** NIST, *Role Based Access Control*, <http://csrc.nist.gov/rbac/rbac-std-ncits.pdf>,
392 Proposed ANSI Standard, BSR INCITS 359, 4/4/2003.
393 **[RBACIntro]** D. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli, *Proposed*
394 *NIST Standard for Role-Based Access Control*,
395 <http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf>, ACM Transaction on Information and
396 System Security, Vol. 4, No. 3, August 2001, pages 224-274.
397 **[XACMLIntro]** A Brief Introduction to XACML, [http://www.oasis-](http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html)
398 [open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html](http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html), 14
399 March 2003.

400 A. Acknowledgments

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

- 403 • *Frank Siebenlist, Argonne National Laboratory*
- 404 • *Daniel Engovatov, BEA Systems, Inc.*
- 405 • *Hal Lockhart, BEA Systems, Inc.*
- 406 • *Tim Moses, Entrust*
- 407 • *Maryann Hondo, IBM*
- 408 • *Michiharu Kudo, IBM*
- 409 • *Michael McIntosh, IBM*
- 410 • *Anthony Nadalin, IBM*
- 411 • *Rebekah Lepro, NASA*
- 412 • *Steve Anderson, OpenNetwork*
- 413 • *Simon Godik, Overxeer*
- 414 • *Bill Parducci, Overxeer*
- 415 • *Anne Anderson, Sun Microsystems*
- 416 • *Seth Proctor, Sun Microsystems*
- 417 • *Polar Humenn, Syracuse University*
- 418 *In addition, the following people made contributions to this specification:*
- 419 • *Ravi Sandhu, George Mason Univ.*
- 420 • *John Barkley, NIST*
- 421 • *Ramaswamy Chandramouli, NIST*
- 422 • *David Ferraiolo, NIST*
- 423 • *Rick Kuhn, NIST*
- 424 • *Serban Gavrilă, VDG Inc.*

425

B. Revision History

426

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

427

C. Notices

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

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

440 **Copyright © OASIS Open 2004. All Rights Reserved.**

441 This document and translations of it may be copied and furnished to others, and derivative works that
442 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published
443 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright
444 notice and this paragraph are included on all such copies and derivative works. However, this document
445 itself does not be modified in any way, such as by removing the copyright notice or references to OASIS,
446 except as needed for the purpose of developing OASIS specifications, in which case the procedures for
447 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required
448 to translate it into languages other than English.

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

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