

1

# XACML Use Case for 2 XML Fine-grained Access Control

3 March 10, 2002

4 Author: Michiharu Kudo

5 This document describes an XACML use case for fine-grained XML resource protection. A couple  
6 of policy specification examples are specified using future XACML extension feature. The policy  
7 specification is based on the latest XACML language proposal document [1] and SAML  
8 specification [2].

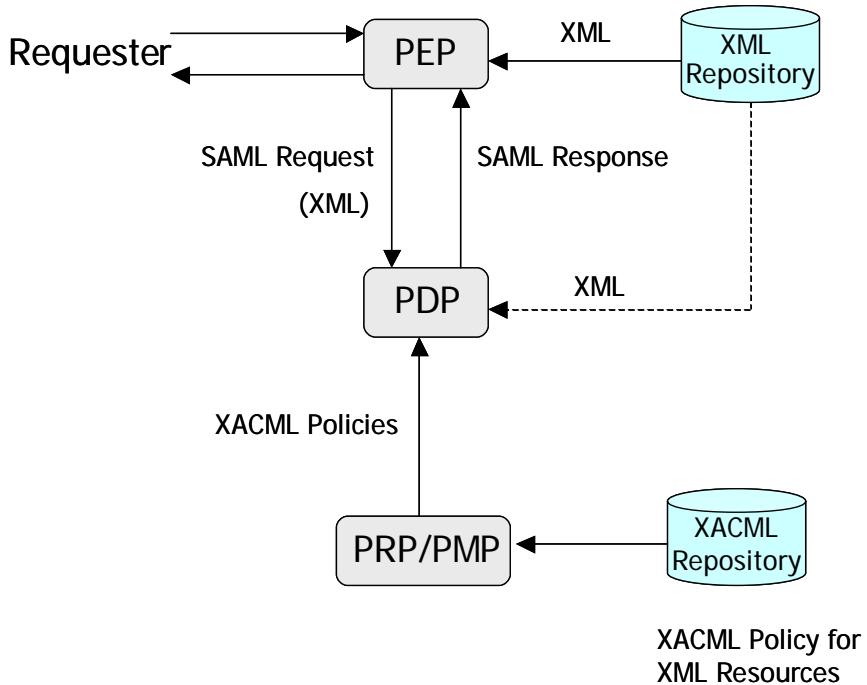
9

## 1. Overview of XML Fine-grained Access Control

10

### 1.1 Data-flow

11 Figure 1 shows a data-flow diagram for XML fine-grained access control.



21 Figure 1 Data-flow diagram

22 Since target resource is XML, it is stored in XML repository (file system, etc.). The target XML  
23 may be sent to PDP by embedding in SAML request, accessed directly by PDP (or PIP), or may not  
24 be referred.

## 25 1.2 Language Primitives

26 Figure 2 shows fundamental XACML language components used for XML fine-grained access  
27 control.

### 28 Policy

```
29 <policyStatement>
  policyId: P1
  metaPolicy: meta-policy-1
  effect: permitIf
  target: (/medico.com/patient*.xml, *, read)
  rules: R1, R2, R3
```

31 Exchangable

```
32 <policyStatement>
  policyId: P1
  metaPolicy: meta-policy-1
  effect: permitIf
  target: (/medico.com/patient*.xml, *, read)
  condition:
```

### 35 Meta-Policy

```
36 <metaPolicyStatement>
  metaPolicyId: meta-policy-1
  description: Global-deny (permitIf, denyIf)
  semantics: <and><or><policyRef> ...
```

```
37 <metaPolicyStatement>
  metaPolicyId: user's meta-policy
  description: Hierarchy-based conflict
    resolution policy, denial-takes precedence
    (positive, negative)
  semantics: http://www.alice.com/mypolicy
```

### Rule

```
<ruleStatement>
  ruleId: R1
  metaPolicy: meta-policy-1
  effect: permitIf
  target: (/phone, Staff, read)
  condition:
```

```
<ruleStatement>
  ruleId: R2
  metaPolicy: meta-policy-1
  effect: permitIf
  target: (/*, Doctor, read)
  condition:
```

```
<ruleStatement>
  ruleId: R3
  metaPolicy: meta-policy-1
  effect: denyIf
  target: (/insurer/phone, Staff, read)
  condition:
```

40

41 Figure 2. Fundamental XACML language components

## 42 2. XML Resource Reference

43 From XML resource protection viewpoint, it is important to define which granularity of the XML  
44 resource can be referred by the access control policy. We consider three kinds of reference methods:

45    1. XPath  
46    2. Simple path expression  
47    3. ID reference

48    XPath resource reference is that any resource units that can be specified using XPath are possible  
49    resource unit. Elements and attributes are typical reference unit. Simple path expression is similar to  
50    the usual path expression used in file systems (UNIX etc.). The difference between XPath and the  
51    simple path expression is briefly described in Section 2.5. ID reference is based on the data  
52    type-based reference method. Both DTD and XML Schema support this ID notion.

53    In this section, we only give a couple of authorization request examples of SAML Request /  
54    Response. Corresponding access control policy examples are described in the next section. We are  
55    assuming that the PEP sends a SAML authorization decision request to the PDP after the PEP  
56    identifies a target XML resource in response to the resource request from the request initiator. We  
57    do not consider cases such that multiple XML resources are referred in the SAML authorization  
58    decision request.

## 59    2.1    **XPath Resource Reference**

### 60    2.1.1    **Access to a specific node of a specific XML file**

61    This example shows a case that a requester needs to access a specific node of XML document. For  
62    example, a hospital staff may need to read a patient's date of birth stored in <patientDoB> element  
63    of the patient-123.xml. Target resource reference is encoded in the SAML  
64    AuthorizationDecisionQuery as follows:

```
65    <AuthorizationDecisionQuery Resource="http://medico.com/patient-123.xml#xpointer(/record/patient/patientDoB)">  
66    ...  
67    </AuthorizationDecisionQuery>
```

69    Target resource reference uses an XPointer syntax [3]<sup>1</sup>. The string before “#” character indicates  
70    URI [4] of the target XML resource and the string after “#xpointer(“ indicates XPath expression [5].  
71    In this example, the URI is “http://medico.com/patient-123.xml” and the XPath is  
72    “/record/patient/patientDoB”. In this case, this XPath refers to only one node (element). Refer to  
73    Section 2.1.3 where XPath expression refers to a set of nodes.

74    In the above example, we assumed that each element of the target XML is uniquely identified  
75    without namespace identifier. When namespace identifier is necessary, the XPath expression looks  
76    like:

```
77    <AuthorizationDecisionQuery      Resource="http://medico.com/patient-123.xml#xmlins      (x=http://medico.com)  
78    xpointer(x:record/x:patient/x:patientDoB)">  
79    ...  
80    </AuthorizationDecisionQuery>
```

---

<sup>1</sup> XACML uses a subset of XPointer specification that allows only XPath expression in “#xpointer()” function. The status of the latest XPointer specification is a candidate recommendation.

82 The above specification means that each element in the patient-123.xml is accompanied with the  
83 namespace URI “<http://medico.com>”.

84 The URI argument can be an arbitrary URI syntax such as  
85 “<file:///c:/winnt/system32/wmpscheme.xml>”. In this case, the PDP and the PEP must have the  
86 identical meaning about the location of the target resource.  
87

```
88 <AuthorizationDecisionQuery Resource=" file:///c:/winnt/system32/wmpscheme.xml#xpointer(/mediaindexscheme)">  
89 ...  
90 </AuthorizationDecisionQuery>  
91
```

92 SAML response looks like:

```
93 <Response xmlns="...draft-sstc-schema-protocol-24.xsd" xmlns:saml=" ...draft-sstc-schema-assertion-24.xsd" >  
94     <saml:Assertion>  
95         <saml:AuthorizationDecisionStatement  
96             Resource=" "http://medico.com/patient-123.xml#xpointer\(/record/patient/patientDoB\)" Decision="Permit">  
97             <saml:Subject>  
98                 <saml:Nameldentifier SecurityDomain="foo" Name="baa"/>  
99             </saml:Subject>  
100            <saml:Actions>  
101                <saml:Action Namespace="http://www.oasis-open.org/.../xmlactions">Read</saml:Action>  
102            </saml:Actions>  
103        </saml:AuthorizationDecisionStatement>  
104    </saml:Assertion>  
105  </Response>  
106
```

## 107 2.1.2 Access to an XML document embedded in SAML request

108 This example shows the case where the target XML resource is embedded in the SAML request as  
109 well as the SAML authorization decision query.

```
110 <AuthorizationDecisionQuery Resource="xpointer(/record/patient/patientDoB)">  
111 ...  
113 </AuthorizationDecisionQuery>
```

114 The URI description is not in the Resource attribute. This means that the target XML is embedded  
115 in the SAML request. The example below shows how the SAML authorization decision request  
116 carries the target XML document. The target XML is indicated in blue color that is inserted in the  
117 ResourceStatement element under the AuthorizationDecisionQuery/Evidence/Assertion element in  
118 the SAML request.  
119

```
120 <Request xmlns="...draft-sstc-schema-protocol-24.xsd" xmlns:saml=" ...draft-sstc-schema-assertion-24.xsd" >  
121     <AuthorizationDecisionQuery  
122         Resource="xmlns(md=medico.com/records.xsd)  
123         xpointer(/md:record/md:patient/md:patientDoB)">  
124         <saml:Subject>  
125             <saml:Nameldentifier SecurityDomain="foo" Name="baa"/>  
126         </saml:Subject>  
127         <saml:Actions>  
128             <saml:Action Namespace="http://www.oasis-open.org/.../xmlactions">Read</saml:Action>  
129         </saml:Actions>  
130         <saml:Evidence>  
131             <saml:Assertion AssertionID="12345" ...>  
132             <saml:ResourceStatement xmlns:samle=" ...samleExt.xsd">  
133                 <md:record xmlns:md="medico.com/records.xsd">  
134                     <md:patientName>  
135                         <md:first>Bartholomew</md:first>
```

```

136         <md:last>Simpson</md:last>
137     </md:patientName>
138     ...
139   </md:record>
140
141   </samle:ResourceStatement>
142   </saml:Assertion>
143   </saml:Evidence>
144 </AuthorizationDecisionQuery>
145 </Request>
```

146 The PDP deals with the XPath of the target resource reference as if the root element of the target  
 147 resource were located just below the samle:ResourceStatement element. The  
 148 samle:ResourceStatement element is an extension point that allows any schema below it.

#### 149 2.1.3 Access to a set of nodes in an XML document

150 There are some cases where the XPath expression of the target resource reference refers to more  
 151 than one node. We allow this flexibility by returning multiple access decision assertions in response  
 152 to the set of nodes requested.

```

153
154 <AuthorizationDecisionQuery Resource="http://medico.com/patient-123.xml#xpointer(//email)">
155 ...
156 </AuthorizationDecisionQuery>
```

157 The above query is asking access decisions for every email elements in the patient-123.xml. Since  
 158 there are four email elements, the resultant SAML response includes four authorization decision  
 159 statements (we omitted namespace prefix “md:” in the Resource attribute for brevity):

```

160
161 <Response xmlns="...draft-sstc-schema-protocol-24.xsd" xmlns:saml=" ...draft-sstc-schema-assertion-24.xsd" >
162   <saml:Assertion>
163     <saml:AuthorizationDecisionStatement          Resource="xpointer(/record/patient/patientContact/email)">
164       Decision="Permit">
165       <saml:Subject>
166         <saml:NameIdentifier SecurityDomain="foo" Name="baa"/>
167       </saml:Subject>
168       <saml:Actions>
169         <saml:Action Namespace="http://www.oasis-open.org/.../xmlactions">Read</saml:Action>
170       </saml:Actions>
171     </saml:AuthorizationDecisionStatement
172
173     <saml:AuthorizationDecisionStatement          Resource=""
174       xpointer(/record/patientGuardian/patientGuardianContact/email)" Decision="Permit">
175       <saml:Subject>
176         <saml:NameIdentifier SecurityDomain="foo" Name="baa"/>
177       </saml:Subject>
178       <saml:Actions>
179         <saml:Action Namespace="http://www.oasis-open.org/.../xmlactions">Read</saml:Action>
180       </saml:Actions>
181     </saml:AuthorizationDecisionStatement
182
183     <saml:AuthorizationDecisionStatement          Resource=""
184       xpointer(/record/primaryCarePhysician/physicianContact/email)" Decision="Deny">
185       <saml:Subject>
186         <saml:NameIdentifier SecurityDomain="foo" Name="baa"/>
187       </saml:Subject>
188       <saml:Actions>
189         <saml:Action Namespace="http://www.oasis-open.org/.../xmlactions">Read</saml:Action>
190       </saml:Actions>
191     </saml:AuthorizationDecisionStatement
192
```

```

193 <saml:AuthorizationDecisionStatement Resource="xpointer(/record/insurer/email)" Decision="Deny">
194   <saml:Subject>
195     <saml:NameIdentifier SecurityDomain="foo" Name="baa"/>
196   </saml:Subject>
197   <saml:Actions>
198     <saml:Action Namespace="http://www.oasis-open.org/.../xmlactions">Read</saml:Action>
199   </saml:Actions>
200 </saml:AuthorizationDecisionStatement>
201 </Assertion>
202 </Request>
```

203 In similar way, it is possible to submit a query about every node included in a specific sub-tree. The  
 204 query in the next example shows an access request to every node in the medical information in the  
 205 medical-123.xml.

```

206
207 <AuthorizationDecisionQuery Resource="http://medico.com/patient-123.xml#xpointer(/record/medical//*)">
208 ...
209 </AuthorizationDecisionQuery>
```

## 210 2.1.4 Access to a set of XML document

211 There are some cases where the target XML resource is not only one XML document. We allow this  
 212 flexibility.

```

213
214 <AuthorizationDecisionQuery
215 Resource="http://medico.com/records#xpointer(/record/patient[@patientName/first='Alice']/patientDoB)">
216 ...
217 </AuthorizationDecisionQuery>
```

218 Above query assumes that the resource URI “<http://medico.com/records>” means a collection of  
 219 XML documents which schema is “[medico.com/records.xsd](http://medico.com/records.xsd)”. If the resource URI contains more  
 220 than one document, the PDP and the PEP must share the notation to indicate the unique identifier of  
 221 the document. For example, if there are two XML documents, ID1212 and ID1212, then the SAML  
 222 response looks like:

```

223 <Response xmlns="...draft-sstc-schema-protocol-24.xsd" xmlns:saml="...draft-sstc-schema-assertion-24.xsd" >
224   <saml:Assertion>
225     <saml:AuthorizationDecisionStatement
226       Resource=" http://medico.com/records/ID1212#xpointer(/record/patient/patientDoB)" Decision="Permit">
227       <saml:Subject>
228         <saml:NameIdentifier SecurityDomain="foo" Name="baa"/>
229       </saml:Subject>
230       <saml:Actions>
231         <saml:Action Namespace="http://www.oasis-open.org/.../xmlactions">Read</saml:Action>
232       </saml:Actions>
233     </saml:AuthorizationDecisionStatement>
234   </saml:Assertion>
235 <saml:AuthorizationDecisionStatement
236   Resource=" http://medico.com/records/ID1234#xpointer(/record/patient/patientDoB)" Decision="Deny">
237   <saml:Subject>
238     <saml:NameIdentifier SecurityDomain="foo" Name="baa"/>
239   </saml:Subject>
240   <saml:Actions>
241     <saml:Action Namespace="http://www.oasis-open.org/.../xmlactions">Read</saml:Action>
242   </saml:Actions>
243 </saml:AuthorizationDecisionStatement>
244   </saml:Assertion>
245 </Response>
```

247 In the above example, we assumed that the first authorization decision statement is the decision to  
248 the document which identifier is ID1212, and that the second authorization decision statement is the  
249 decision to the document which identifier is ID1234. This mapping definition depends on each  
250 application.

251 **2.2 Simple Path Resource Reference**

252 Potential problem of using XPath expression to point target node(s) would be that the PDP must  
253 retrieve the target XML instance document from the XML repository because most of the  
254 commercial XPath processors require XML instance document in evaluating the XPath expression.  
255 If the target XML is huge, it is not wise to retrieve each XML instance. Then, we use a simple path  
256 expression instead of rich XPath expression. The simple path expression is based on popular path  
257 expressions used in e.g. file systems that just uses “/” as a path separator.

258   <AuthorizationDecisionQuery Resource="http://medico.com/records?/record/patient/patientDoB">  
259   ...  
260   </AuthorizationDecisionQuery>

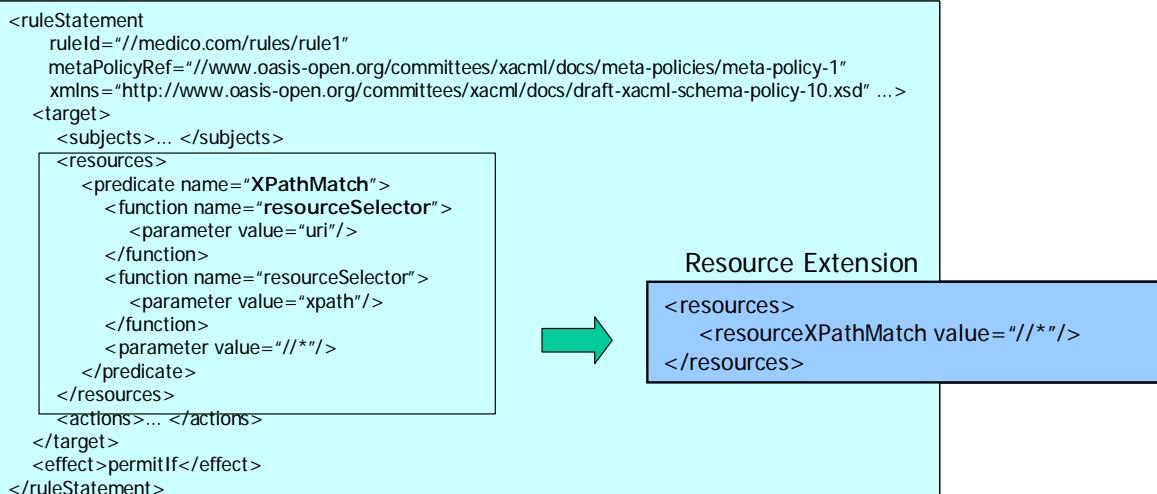
262 Above query has a simple path expression “/record/patient/patientDoB”. This path expression may  
263 satisfy the path expression written in the policy like “/record/\*/patientDoB”. We do not specify here  
264 the syntax and the semantics of the simple path expression in detail.  
265

### 3. XACML Policies and Resource Extensions

This section shows two XACML extension examples that make policy specifications much more concise and facilitate policy authoring.

#### 3.1 Example 1: Resource Extension using XPath

Figure 3 shows how to write resource-dependent functions and predicates. XPath expression is used in the resource attribute in SAML authorization decision request as Figure 3 shows.



Predicate : **XPathMatch**(arg1,arg2,arg3)  
Return : Boolean  
arg1 : String (URI)  
arg2 : String (XPath)  
arg3 : String (XPath)

Function : **resourceSelector**(returnType)  
Return : String  
returnType : ["uri", "path", "xpath", or "all"]

e.g.  
XPathMatch("medico.com/patient-123.xml",  
"/record/patient/patientDoB", "//\*")  
→ true

e.g.  
resourceSelector("//medico.com/patient-123.xml"  
"#xpointer(/record/patient/patientDoB)")  
→ /record/patient/patientDoB

Figure 3. Resource extension using XPath expression

In the box of resource extension, `<resourceXPathMatch>` element indicates a potential resource extension (or macro) that provides the same meaning as the left-hand predicate-based specification. The merit of using resource extension is to make policy much more concise and comprehensible to human users.

## 3.2 Example2: Resource Extension using Simple Path Expression

Figure 4 shows how the simple path expression (refer to Section 2.5) is used in the resource attribute in SAML authorization decision request.

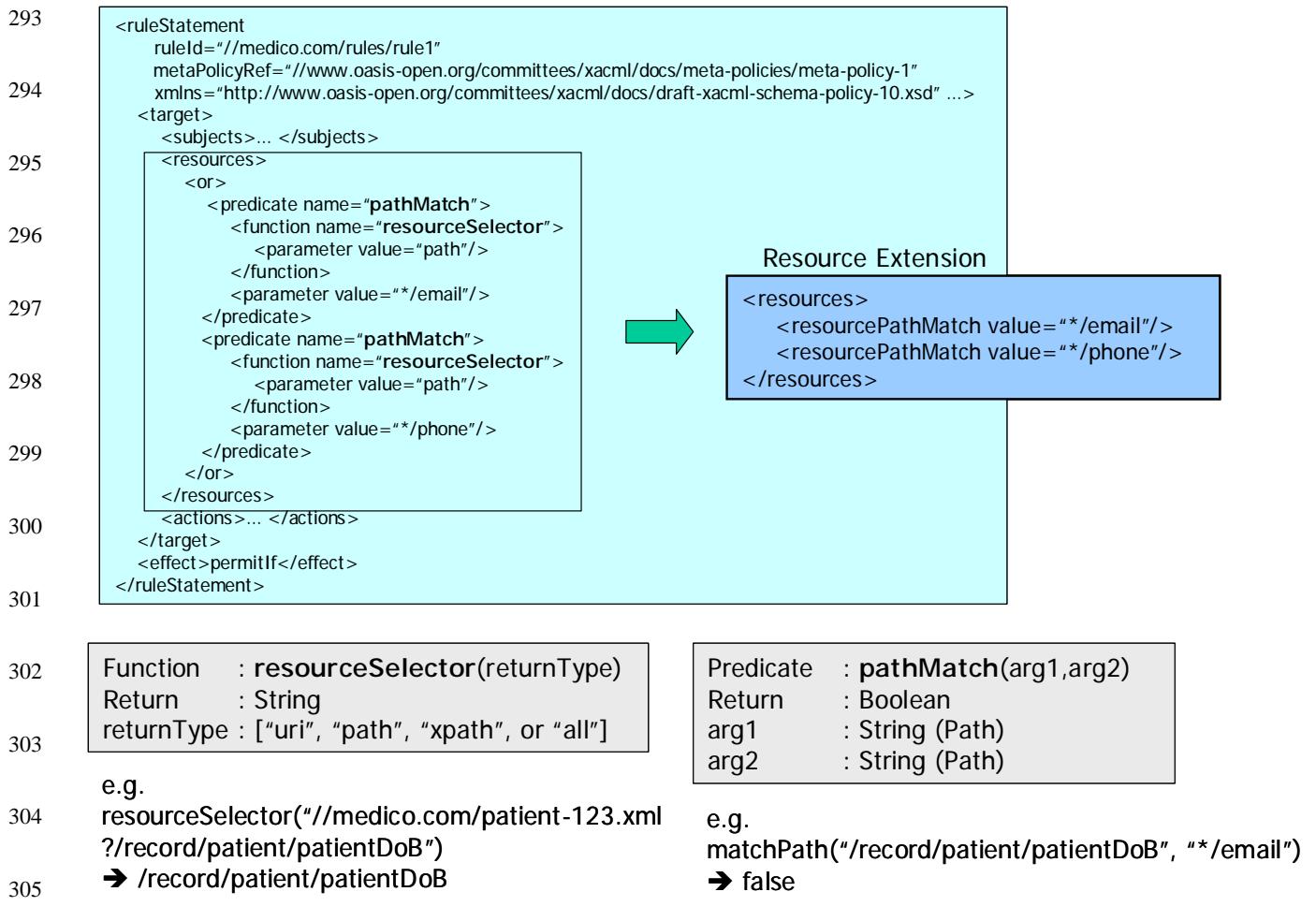
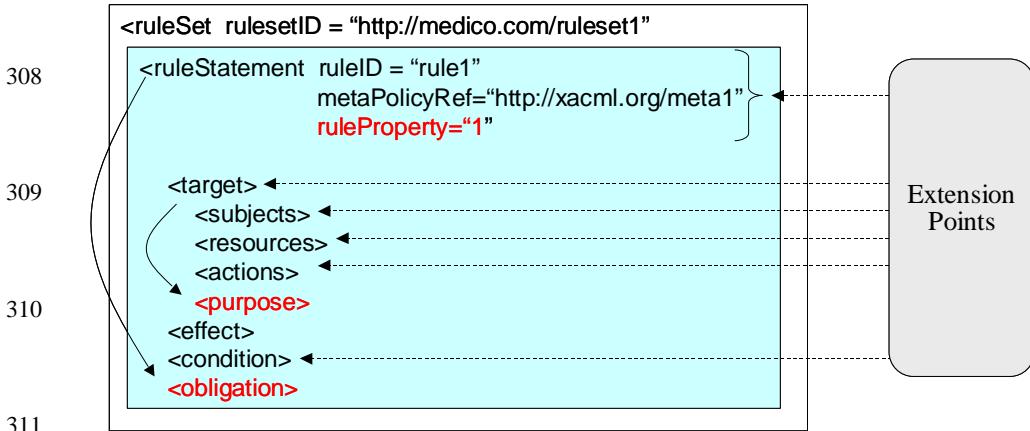


Figure 4. Resource extension using simple path expression.

307    **3.3 Potential Extension Points**



312    Figure 5. Potential Extension Points

313

## 4. XACML Policy Evaluation

In this section, we describe two access control policies that use different semantics of policy evaluation. The first policy uses XACML pre-defined meta policy “meta-policy-1”. The second policy uses a user defined hierarchy-based meta policy “meta-policy-hierarchy”. It shows how user defined meta-policy is incorporated in XACML policy rules.

### 4.1 Policy Evaluation Semantics using Meta-Policy-1

Figure 6 shows overview of XACML policy evaluation flow in PDP.

1. SAML access request

```
resource : //medico.com/patient-123.xml#xpointer(/record/patient/patientDoB)
subject : Alice (Doctor)
Action : read
```

2. Find Applicable Policy

```
<policySet>
<policyStatement>
  policyId: P1
  (metaPolicy: meta-policy-1)
  effect: permitIf
  target: (/medico.com/patient*.xml, *, read)
  rules: R1, R2, R3 (RuleSet1)
```

3. P1 is applicable

```
ruleset1.xml
<ruleSet>
  ruleId: RuleSet1
  metaPolicy: meta-policy-1
  <ruleStatement>
    ruleId: R1
    metaPolicy: meta-policy-1
    effect: permitIf
    target: (/phone, Operator, -)
  </ruleStatement>
  <ruleStatement>
    ruleId: R2
    metaPolicy: meta-policy-1
    effect: permitIf
    target: (/*, Doctor, -)
  </ruleStatement>
  <ruleStatement>
    ruleId: R3
    metaPolicy: meta-policy-1
    effect: denyIf
    target: (/insurer/phone, Operator, -)
```

4. Find rules

5. “R2” becomes true → permitIf

6. metaPolicy evaluation → permit

7. SAML access response

Decision: Permit

```
<metaPolicyStatement>
  metaPolicyId: meta-policy-1
  description: Global-deny (permitIf, denyIf)
  semantics: <and><or><policyRef> ...
```

Physical component unit

Figure 6. Access evaluation flow using Meta-Policy-1

The following descriptions explain the evaluation process of the PDP.

**Step 1.** The PDP receives SAML authorization decision request. The request is to access a patientDoB element in the http://medico.com/patient-123.xml file. This request is equivalent to the request described in the example 3.1.

- 337   **Step 2.** PRP/PAP retrieves a set of policies that is applicable to the access request. Target in  
338   policyStatement is used to select an applicable policy using just checking parameters in SAML  
339   access request.
- 340   **Step 3.** The PRP/PAP finds “P1” is an applicable policy.
- 341   **Step 4.** The PRP/PAP finds that the P1’s subsidiary rules are “R1”, “R2” and “R3”, which are stored  
342   in ruleset1.xml. The PRP/PAP send them to the PDP.
- 343   **Step 5.** The PDP evaluates a set of rules and finds that “R2” becomes true. The PDP computes the  
344   result based on the meta-policy “meta-policy-1”. In the case where Staff submits an access request  
345   on /patient/issuer/phone element, the target parts of “R1” and “R3” are satisfied. The effect of the  
346   R1 is “PermitIf” and the effect of the R3 is “DenyIf”. Since the meta-policy-1 means denials always  
347   take precedence over permits, the final decision is determined as “Deny”.
- 348   **Step 6.** Since the R2’s effect is “permitIf” and there is no rule that holds, the meta-policy-1  
349   generates “Permit” as a final decision.
- 350   **Step 7.** The PDP generates a SAML authorization decision assertion.

## 4.2 Policy Evaluation Semantics using Meta-Policy-Hierarchy

Figure 7 shows overview of XACML policy evaluation flow in PDP.

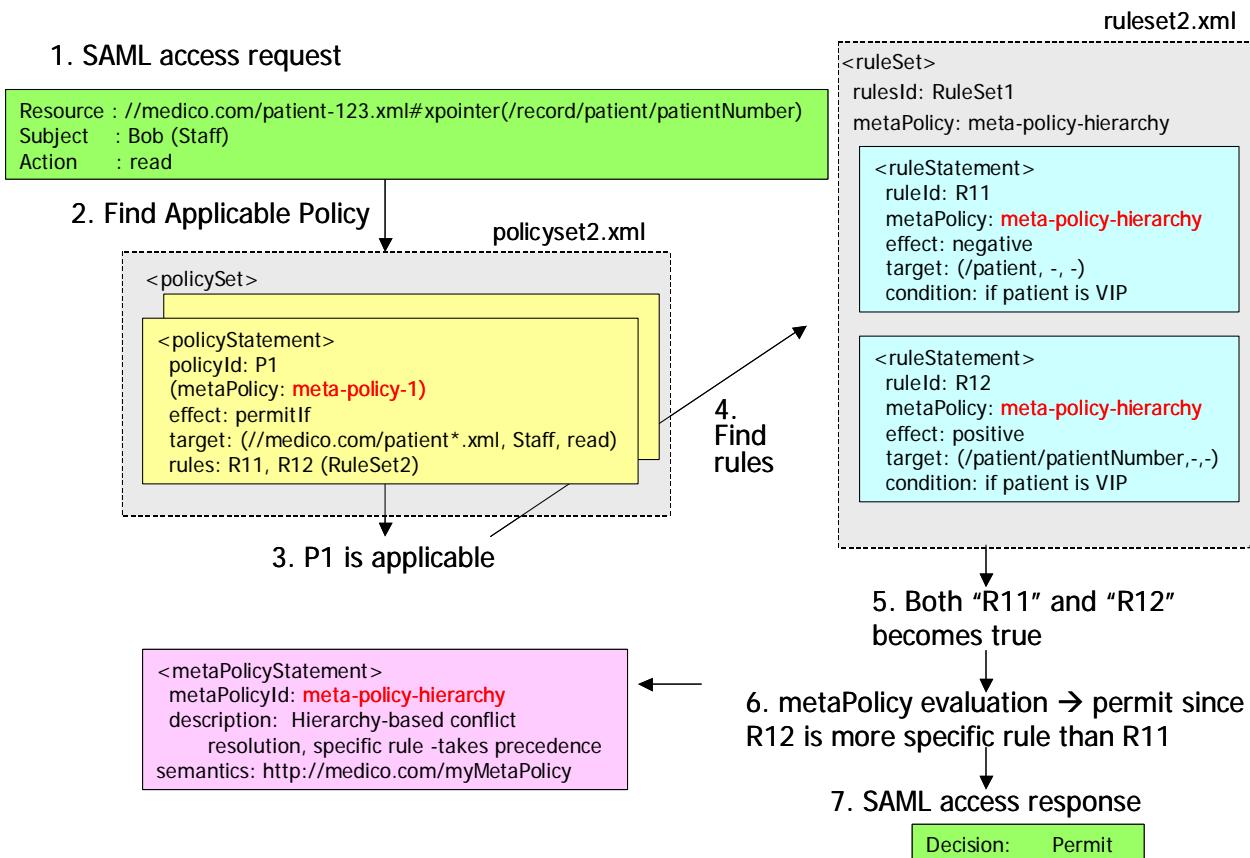


Figure 7. Access evaluation flow using Meta-Policy-Hierarchy

The following descriptions explain the evaluation process of the PDP.

**Step 1.** The PDP receives SAML authorization decision request. The request is to access a patientNumber element in the http://medico.com/patient-123.xml file.

**Step 2.** PRP/PAP retrieves a set of policies that is applicable to the access request. Target in policyStatement is used to select an applicable policy using just checking parameters in SAML access request.

**Step 3.** The PRP/PAP finds that “P1” is an applicable policy.

**Step 4.** The PRP/PAP finds that the P1’s subsidiary rules are “R11” and “R12”, which are stored in ruleset2.xml. The PRP/PAP send them to the PDP.

**Step 5.** The PDP evaluates a set of rules and finds that both the R11 and the R12 become true since the patient is VIP. Both rules are conflicting because the R11 denies while the R12 grants,

376   **Step 6.** Then the PDP calls the user-defined meta-policy “meta-policy-hierarchy” by URI. The  
377   meta-policy determines grant decision because the rule R12 is more specific rule than the R11.

378   **Step 7.** The PDP generates a grant SAML authorization decision assertion.

380   

## Reference

381   [1] XACML, The OASIS extensible Access Control Markup Language (XACML), Committee  
382   Draft Version 1.0, 8 March, 2002, <http://www.oasis-open.org/committees/xacml>

383   [2] SAML, The OASIS Assertions and Protocol for the OASIS Security Assertion Markup  
384   Language (SAML), Committee Working Draft, Version 27, February 14<sup>th</sup>, 2002, ,  
385   <http://www.oasis-open.org/committees/security>

386   [3] XPointer, XML Pointer Language (XPointer) Version 1.0, W3C Candidate Recommendation, 11  
387   September 2001, <http://www.w3.org/TR/2001/CR-xptr-20010911>.

388   [4] URI, RFC 2396.

389   [5] XPath, XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999,  
390   <http://www.w3.org/TR/xpath>.