



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23

The OASIS XML-based Access- Control Markup Language (XACML)

Document identifier: draft-xacml-v0.10

Location: <http://www.oasis-open.org/committees/xacml/docs>

Publication date: 8 Mar 2002

Maturity Level: Committee Draft

Send comments to: xacml-comment@lists.oasis-open.org

Editors:

Tim Moses, Entrust
Simon Godik, Simon Godik

Contributors:

Anne Anderson, Sun Microsystems
Bill Parducci, Bill Parducci
Carlisle Adams, Entrust
Ernesto Damiani, University of Milan
Hal Lockhart, Entegritiy
Ken Yagen, Crosslogix
Michiharu Kudo, IBM, Japan
Pierangela Samarati, University of Milan
Polar Humenn, University of Syracuse
Sekhar Vajjhala, Sun Microsystems

24	Table of contents	
25	THE OASIS XML-BASED ACCESS-CONTROL MARKUP LANGUAGE (XACML)	1
26	1. GLOSSARY (NON-NORMATIVE)	5
27	1.1. Preferred terms	5
28	1.2. Related terms	6
29	2. INTRODUCTION (NON-NORMATIVE)	6
30	2.1. Notation	6
31	2.2. Schema Organization and Namespaces	7
32	3. EXAMPLE (NON-NORMATIVE)	7
33	3.1. Introduction to the example	7
34	3.2. Example medical record instance	7
35	3.3. Example authorization decision request	9
36	3.4. Example plain-language rules	10
37	3.5. Example XACML rule instances	10
38	3.5.1. Rule 1	10
39	3.5.2. Rule 2	11
40	3.5.3. Rule 3	13
41	3.5.4. Rule 4	14
42	4. MODELS (NON-NORMATIVE)	15
43	4.1. Data-flow model	15
44	4.2. Rule combination	17
45	4.2.1. Policy Statement	17
46	4.2.2. Rule statement	18
47	4.3. Policy language model	20
48	4.3.1. Target	23
49	5. POLICY SYNTAX (NORMATIVE, WITH THE EXCEPTION OF THE SCHEMA	
50	FRAGMENTS)	23
51	5.1. Element <policyStatement>	23

52	5.2.	Element <ruleStatement>	24
53	5.3.	Complex type PolicyStatementType	24
54	5.4.	Complex type RuleStatementType	24
55	5.5.	Complex type TargetType	24
56	5.6.	Complex type SubjectType	25
57	5.7.	Complex type ResourceType	25
58	5.8.	Complex type ActionType	25
59	5.9.	Simple type EffectType	25
60	5.10.	Complex type PredicateExpressionType	25
61	5.11.	Element <predicateExpression>	26
62	5.12.	Complex type PredicateExpressionAbstractType	26
63	5.13.	Element <predicate>	26
64	5.14.	Complex type PredicateAbstractType	26
65	5.15.	Element <and>	26
66	5.16.	Element <or>	26
67	5.17.	Element <orderedOr>	26
68	5.18.	Element <nOf>	26
69	5.19.	Element <not>	27
70	5.20.	Complex type AndType	27
71	5.21.	Complex type OrType	27
72	5.22.	Complex type OrderedOrType	27
73	5.23.	Complex type NofType	27
74	5.24.	Complex type NotType	28
75	5.25.	Element <present>	28
76	5.26.	Element <equal>	28
77	5.27.	Element <greaterOrEqual>	28
78	5.28.	Element <lessOrEqual>	28
79	5.29.	Element <subset>	29

80	5.30.	Element <superset>	29
81	5.31.	Element <patternMatch>	29
82	5.32.	Element <nonNullSetInterscetion>	29
83	5.33.	Complex type PresentType	29
84	5.34.	Complex type CompareType	29
85	5.35.	Complex type RuleRefType	30
86	6.	XACML IDENTIFIERS (NORMATIVE)	30
87	6.1.	Requestor	30
88	6.2.	Time of day	30
89	6.3.	Authentication locality	30
90	6.4.	Meta-policy one	30
91	7.	META-POLICY ONE (NON-NORMATIVE)	30
92	8.	PROFILES (NON-NORMATIVE)	31
93	8.1.	XACML	31
94	8.2.	SAML	31
95	8.3.	XML Digital Signature	31
96	8.4.	LDAP	31
97	9.	XACML EXTENSION POINTS (NON-NORMATIVE)	31
98	10.	SECURITY AND PRIVACY (NON-NORMATIVE)	31
99	11.	REFERENCES	32
100	12.	SCHEMA (NORMATIVE)	32
101	13.	CONFORMANCE (NORMATIVE)	35
102		APPENDIX A. NOTICES	37
103			

104

105 **1. Glossary (non-normative)**

106 **1.1. Preferred terms**

107 *Access* - Performing an *action* on a *resource*

108 *Access control* - Controlling *access* in accordance with a *policy*

109 *Action* - An operation that may be performed on a *resource*

110 *Applicable policy* - The complete set of *rules* that governs *access* for a specific *decision request*

111 *Attribute* - Characteristic of a *subject, resource, action, environment or rule* that may be referenced by a
112 *rule*

113 *Authorization decision* - The result of evaluation of an *applicable policy*. A function that evaluates to
114 "permit, deny or indeterminate", and (optionally) a set of *post-conditions*

115 *Condition* - An expression of *predicates* over *attributes* of *resource, subject* and *environment*

116 *Context* - The intended use of information revealed as a result of *access*.

117 *Decision request* - The request by a *PEP* to a *PDP* to render an *authorization decision*

118 *Environment* - The set of *attributes* that may be referenced by *rules* and that are independent of a particular
119 *subject, resource, action or rule*

120 *Information request* - The request by a *PDP* to a *PIP* for one or more *environment attributes*

121 *Meta-policy* - A procedure for combining *rules* to form a *policy*

122 *Meta-policy one* - A *meta-policy* defined in the current version of XACML

123 *Policy* - A set of *rules* and an identifier for the *meta-policy* for combining the *rules*

124 *Policy administration point (PAP)* - The system entity that creates a *rule*

125 *Policy decision point (PDP)* - The system entity that evaluates *applicable policy* and renders an
126 *authorization decision*

127 *Policy enforcement point (PEP)* - The system entity that performs *access control*, by enforcing
128 *authorization decisions*

129 *Policy information point (PIP)* - The system entity that acts as the source of *environment attributes*

130 *Policy mediation point (PMP)* - The system entity that resolves conflict amongst *rules*

131 *Policy retrieval point (PRP)* - The system entity that creates *applicable policy* from *rules* by the application
132 of *meta-policy*

133 **Post-condition** - A process specified in a *policy* that must be completed in conjunction with *access*. There are
134 two types of post-condition: an *internal post-condition* must be performed by the *PDP* in conjunction with
135 the issuance of a "permit" response, and an *external post-condition* must be performed by the *PEP* in
136 conjunction with permitting *access*

137 **Predicate** - A statement about *attributes* whose truth can be evaluated

138 **Resource** - Data, service, or system component

139 **Rule** - A *target* and a set of *conditions* with an associated *meta-policy* identifier

140 **Subject** - An actor whose *attributes* may be referenced in a *rule*

141 **Target** - The set of *decision requests*, identified by definitions of *resource*, *subject* and *action* to which a *rule*
142 applies

143 **Target mapping** - The process of confirming that a *rule* is applicable to a given *target*.

144 **1.2. Related terms**

145 In the field of access control and authorization there are several closely related terms in common use. For
146 purposes of precision and clarity, certain of these terms are not used in this specification.

147 For instance, the term *attribute* is used in place of the terms: *group*, *role*, *privilege*, *permission*, *right*,
148 *authorization* and *entitlement*.

149 The term *object* is also in common use, but we use the term *resource* in this specification.

150 *Requestors* and *initiators* are covered by the term *subject*.

151 **2. Introduction (non-normative)**

152 This specification defines the syntax and semantics for XML-encoded access control rule statements and
153 policy statements. The XACML schema is an extension schema for SAML.

154 The following sections describe how to understand the rest of this specification.

155 **2.1. Notation**

156 This specification contains schema conforming to W3C XML Schema and normative text to describe the
157 syntax and semantics of XML-encoded policy statements.

158 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",
159 "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be
160 interpreted as described in IETF RFC 2119 [rfc2119](#):

161 *"they MUST only be used where it is actually required for interoperation or to limit*
162 *behavior which has potential for causing harm (e.g., limiting retransmissions)"*

163 These keywords are thus capitalized when used to unambiguously specify requirements over protocol and
164 application features and behavior that affect the interoperability and security of implementations. When these
165 words are not capitalized, they are meant in their natural-language sense.

166 Listings of XACML schemas appear like this.

167
168

Example code listings appear like this.

169 Conventional XML namespace prefixes are used throughout the listings in this specification to stand for their
170 respective namespaces (see Section 2.2) as follows, whether or not a namespace declaration is present in the
171 example:

- 172 • The prefix `saml`: stands for the SAML assertion namespace.
- 173 • The prefix `ds`: stands for the W3C XML Signature namespace.
- 174 • The prefix `xs`: stands for the W3C XML Schema namespace in example listings. In schema
175 listings, this is the default namespace and no prefix is shown.

176 This specification uses the following typographical conventions in text: `<SAMLElement>`,
177 `<ns:ForeignElement>`, `Attribute`, **Datatype**, `OtherCode`.

178 2.2. Schema Organization and Namespaces

179 The XACML policy statement structures are defined in a schema associated with the following XML
180 namespace:

181 <http://www.oasis-open.org/committees/xacml/docs/draft-xacml-schema-09.xsd>

Comment: Page: 1
This will need to be edited before
final release

182 **Note:** The XACML namespace names are temporary and may change when
183 XACML 1.0 is finalized.

184 The SAML assertion schema is imported into the XACML schema. Also imported is the schema for XML
185 Signature `XMLSigXSD`, which is associated with the following XML namespace:

186 <http://www.w3.org/2000/09/xmldsig#>

187 3. Example (non-normative)

188 3.1. Introduction to the example

189 This section contains an example of the application of XACML rules to a medical record. Four separate rules
190 are defined.

191 3.2. Example medical record instance

192 Following is an instance of a medical record to which the example rules can be applied. The "record" schema
193 is defined in the registered namespace administered by "medico.com".

```
194 <?xml version="1.0" encoding="UTF-8"?>  
195 <record xmlns="medico.com/records.xsd"  
196 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
197 xsi:schemaLocation="medico.com/records.xsd  
198 D:\MYDOCU~1\Standards\XACML\record.xsd">  
199   <patient>  
200     <patientName>  
201       <first>Bartholomew</first>  
202       <last>Simpson</last>  
203     </patientName>  
204     <patientContact>  
205       <street>27 Shelbyville Road</street>  
206       <city>Springfield</city>
```

```

207     <state>MA</state>
208     <zip>12345</zip>
209     <phone>555.123.4567</phone>
210     <fax/>
211     <email/>
212   </patientContact>
213   <patientDoB xsi:type="date">1992-03-21</patientDoB>
214   <patientGender xsi:type="string">male</patientGender>
215   <policyNumber xsi:type="string">555555</policyNumber>
216 </patient>
217 <parentGuardian>
218   <parentGuardianName>
219     <first>Homer</first>
220     <last>Simpson</last>
221   </parentGuardianName>
222   <parentGuardianContact>
223     <street>27 Shelbyville Road</street>
224     <city>Springfield</city>
225     <state>MA</state>
226     <zip>12345</zip>
227     <phone>555.123.4567</phone>
228     <fax/>
229     <email>homers@aol.com</email>
230   </parentGuardianContact>
231 </parentGuardian>
232 <primaryCarePhysician>
233   <physicianName>
234     <first>Julius</first>
235     <last>Hibbert</last>
236   </physicianName>
237   <physicianContact>
238     <street>1 First St</street>
239     <city>Springfield</city>
240     <state>MA</state>
241     <zip>12345</zip>
242     <phone>555.123.9012</phone>
243     <fax>555.123.9013</fax>
244     <email/>
245   </physicianContact>
246   <registrationID>ABC123</registrationID>
247 </primaryCarePhysician>
248 <insurer>
249   <name>Blue Cross</name>
250   <street>1234 Main St</street>
251   <city>Springfield</city>
252   <state>MA</state>
253   <zip>12345</zip>
254   <phone>555.123.5678</phone>
255   <fax>555.123.5679</fax>
256   <email/>
257 </insurer>
258 <medical>
259   <treatment>
260     <drug>
261       <name>methylphenidate hydrochloride</name>
262       <dailyDosage>30mgs</dailyDosage>
263       <startDate>1999-01-12</startDate>
264     </drug>
265     <comment>patient exhibits side-effects of skin coloration
266 and carpal degeneration</comment>
267   </treatment>
268   <result>
269     <test>blood pressure</test>

```


270
271
272
273
274
275

```
<value>120/80</value>
<date>2001-06-09</date>
<performedBy>Nurse Betty</performedBy>
</result>
</medical>
</record>
```

276

3.3. Example authorization decision request

277
278
279

The following example illustrates a SAML authorization decision request to which the example rules are intended to be applicable. It represents a request by Julius Hibbert to read the patient date of birth in the record of Bartholomew Simpson. It includes an authentication assertion and an attribute assertion.

280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328

```
<?xml version="1.0" encoding="UTF-8"?>
<Request RequestID="47823081" MajorVersion="0" MinorVersion="24"
xmlns="http://www.oasis-open.org/committees/security/docs/draft-
sstc-schema-protocol-24.xsd"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:saml="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-
24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-protocol-24.xsd
D:\MYDOCU~1\Standards\XACML\V10SCH~1\draft-sstc-schema-protocol-
24.xsd">
  <AuthorizationDecisionQuery
Resource="/medico.com/record/patient[@patientName/first='Bartholome
w'][@patientName/last='Simpson']/patientDoB">
    <saml:Subject>
      <saml:NameIdentifier SecurityDomain="medico.com"
Name="Julius Hibbert"/>
    </saml:Subject>
    <saml:Actions>
      <saml:Action>read</saml:Action>
    </saml:Actions>
    <saml:Evidence>
      <Assertion AssertionID="64578390" Issuer="medico.com"
IssueInstant="2002-03-08T08:23:47-05:00" MajorVersion="0"
MinorVersion="24" xmlns="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-
24.xsd" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-24.xsd
D:\MYDOCU~1\Standards\XACML\V10SCH~1\draft-sstc-schema-assertion-
24.xsd">
        <AuthenticationStatement AuthenticationInstant="2002-
03-08T08:23:45-05:00" AuthenticationMethod="http://www.oasis-
open.org/committees/security/docs/draft-sstc-core-24/password-shal">
          <Subject>
            <NameIdentifier SecurityDomain="medico.com"
Name="Julius Hibbert"/>
            <SubjectConfirmation>
              <ConfirmationMethod>http://www.oasis-
open.org/committees/security/docs/draft-sstc-core-
24/artifact</ConfirmationMethod>
            </SubjectConfirmation>
          </Subject>
          <AuthenticationLocality IPAddress="217.57.95.242"/>
        </AuthenticationStatement>
      </Assertion>
    </saml:Evidence>
```

```

329     <saml:Evidence>
330         <Assertion MajorVersion="0" MinorVersion="24"
331 AssertionID="68938960" Issuer="medico.com" IssueInstant="2000-06-
332 15T15:02:39-05:00" xmlns="http://www.oasis-
333 open.org/committees/security/docs/draft-sstc-schema-assertion-
334 24.xsd" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
335 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
336 xsi:schemaLocation="http://www.oasis-
337 open.org/committees/security/docs/draft-sstc-schema-assertion-24.xsd
338 D:\MYDOCU~1\Standards\XACML\V10SCH~1\draft-sstc-schema-assertion-
339 24.xsd">
340             <AttributeStatement>
341                 <Subject>
342                     <NameIdentifier SecurityDomain="medico.com"
343 Name="Julius Hibbert"/>
344                 </Subject>
345                 <Attribute AttributeName="role"
346 AttributeNamespace="//medico.com">
347                     <AttributeValue>
348                         <role
349 xmlns="http://www.w3.org/2001/XMLSchema/string">physician</role>
350                     </AttributeValue>
351                 </Attribute>
352             </AttributeStatement>
353         </Assertion>
354     </saml:Evidence>
355 </AuthorizationDecisionQuery>
356 </Request>

```

357 **3.4. Example plain-language rules**

358 The following plain-language rules are to be enforced:

- 359 1. A person may read any record for which he or she is the designated patient.
- 360 2. A person may read any record for which he or she is the designated parent or guardian, and for which the
361 patient is under 16 years of age.
- 362 3. A physician may write any medical element for which he or she is the designated primary care physician.
- 363 4. An administrator may read and write any record during office hours and from a designated IP address
364 sub-space.

365 These rules may be written by different PAPS, operating independently, or by a single PAP. Note that
366 XACML instances are intended to be conveyed in SAML assertions, which provide elements for rule and
367 policy meta-data, such as "issuer" and "issueInstant". However, this detail has been omitted from the
368 following examples.

369 **3.5. Example XACML rule instances**

370 **3.5.1. Rule 1**

371 The following XACML ruleStatement instance expresses Rule 1.

372 `<?xml version="1.0" encoding="UTF-8"?>`

```

373 <ruleStatement ruleId="//medico.com/rules/rule1"
374 metaPolicyRef="//www.oasis-open.org/committees/xacml/docs/meta-
375 policies/meta-policy-1" xmlns="http://www.oasis-
376 open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"
377 xmlns:saml="http://www.oasis-
378 open.org/committees/security/docs/draft-sstc-schema-assertion-
379 24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
380 xsi:schemaLocation="http://www.oasis-
381 open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd
382 D:\MYDOCU~1\Standards\XACML\V10SCH~1\XACMLV~4.XSD">
383   <comment>A person may read any record for which he or she is the
384   designated patient</comment>
385   <target>
386     <subjects>
387       <saml:Attribute AttributeName="RFC822Name"
388 AttributeNamespace="//medico.com">
389         <saml:AttributeValue>
390           <rfc822Name
391 xmlns="http://www.w3.org/2001/XMLSchema/string">*</rfc822Name>
392           </saml:AttributeValue>
393         </saml:Attribute>
394       </subjects>
395       <resources>
396         <saml:Attribute AttributeName="documentURI"
397 AttributeNamespace="//medico.com">
398           <saml:AttributeValue>
399             <documentURI
400 xmlns="http://www.w3.org/2001/XMLSchema/string">//medico.com/records
401             .*</documentURI>
402             </saml:AttributeValue>
403           </saml:Attribute>
404         </resources>
405         <actions>
406           <saml:Actions>
407             <saml:Action>read</saml:Action>
408           </saml:Actions>
409         </actions>
410       </target>
411       <effect>permitIf</effect>
412       <condition>
413         <equal>
414           <saml:AttributeDesignator AttributeName="requestor"
415 AttributeNamespace="//oasis-
416 open.org/committees/xacml/docs/identifiers/"/>
417           <saml:AttributeDesignator AttributeName="patientName"
418 AttributeNamespace="//medico.com/record/patient/"/>
419         </equal>
420       </condition>
421     </ruleStatement>

```

3.5.2. Rule 2

The following XACML ruleStatement instance expresses Rule 2.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

425 <ruleStatement ruleId="//medico.com/rules/rule2"
426 metaPolicyRef="//www.oasis-open.org/committees/xacml/docs/meta-
427 policies/meta-policy-1" xmlns="http://www.oasis-
428 open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"
429 xmlns:saml="http://www.oasis-
430 open.org/committees/security/docs/draft-sstc-schema-assertion-
431 24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
432 xsi:schemaLocation="http://www.oasis-
433 open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd
434 D:\MYDOCU~1\Standards\XACML\V10SCH~1\XACMLV~4.XSD">
435   <comment>A person may read any record for which he or she is the
436   designated parent or guardian, and for which the patient is under 16
437   years of age</comment>
438   <target>
439     <subjects>
440       <saml:Attribute AttributeName="RFC822Name"
441 AttributeNamespace="//medico.com">
442         <saml:AttributeValue>
443           <rfc822Name
444 xmlns="http://www.w3.org/2001/XMLSchema/string">*</rfc822Name>
445         </saml:AttributeValue>
446       </saml:Attribute>
447     </subjects>
448     <resources>
449       <saml:Attribute AttributeName="documentURI"
450 AttributeNamespace="//medico.com">
451         <saml:AttributeValue>
452           <documentURI
453 xmlns="http://www.w3.org/2001/XMLSchema/string">//medico.com/records
454 .*</documentURI>
455         </saml:AttributeValue>
456       </saml:Attribute>
457     </resources>
458     <actions>
459       <saml:Actions>
460         <saml:Action>read</saml:Action>
461       </saml:Actions>
462     </actions>
463   </target>
464   <effect>permitIf</effect>
465   <condition>
466     <and>
467       <equal>
468         <saml:AttributeDesignator AttributeName="requestor"
469 AttributeNamespace="//oasis-
470 open.org/committees/xacml/docs/identifiers/">
471         <saml:AttributeDesignator
472 AttributeName="parentGuardianName"
473 AttributeNamespace="//medico.com/record/parentGuardian/">
474       </equal>
475       <greaterOrEqual>
476         <saml:AttributeDesignator AttributeName="patientDOB"
477 AttributeNamespace="//medico.com/record/patient/">
478         <saml:Attribute AttributeName="dateOfBirth"
479 AttributeNamespace="//medico.com">
480           <saml:AttributeValue>
481             <dateOfBirth>1986-03-08</dateOfBirth>
482           </saml:AttributeValue>
483         </saml:Attribute>
484       </greaterOrEqual>
485     </and>
486   </condition>
487 </ruleStatement>

```

488 Notes:

489 The approach to the patient's age is clearly unsatisfactory as it stands, because it requires the rule to be
490 updated daily with a new date. It will be necessary to include arithmetic operators in the language. Another
491 option is to use an external function for the calculation.

492 3.5.3. Rule 3

493 The following XACML ruleStatement instance expresses Rule 3.

```
494 <?xml version="1.0" encoding="UTF-8"?>
495 <ruleStatement ruleId="//medico.com/rules/rule3"
496 metaPolicyRef="//www.oasis-open.org/committees/xacml/docs/meta-
497 policies/meta-policy-1" xmlns="http://www.oasis-
498 open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"
499 xmlns:saml="http://www.oasis-
500 open.org/committees/security/docs/draft-sstc-schema-assertion-
501 24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
502 xsi:schemaLocation="http://www.oasis-
503 open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd
504 D:\MYDOCU~1\Standards\XACML\V10SCH~1\XACMLV~4.XSD">
505   <comment>A physician may write any medical element for which he
506   or she is the designated primary care physician</comment>
507   <target>
508     <subjects>
509       <saml:Attribute AttributeName="role"
510 AttributeNamespace="//medico.com">
511         <saml:AttributeValue>
512           <role
513 xmlns="http://www.w3.org/2001/XMLSchema/string">physician</role>
514         </saml:AttributeValue>
515       </saml:Attribute>
516     </subjects>
517     <resources>
518       <saml:Attribute AttributeName="documentURI"
519 AttributeNamespace="//medico.com">
520         <saml:AttributeValue>
521           <documentURI
522 xmlns="http://www.w3.org/2001/XMLSchema/string">//medico.com/records
523 /medical.*</documentURI>
524         </saml:AttributeValue>
525       </saml:Attribute>
526     </resources>
527     <actions>
528       <saml:Actions>
529         <saml:Action>write</saml:Action>
530       </saml:Actions>
531     </actions>
532   </target>
533   <effect>permitIf</effect>
534   <condition>
535     <equal>
536       <saml:AttributeDesignator AttributeName="requestor"
537 AttributeNamespace="//oasis-
538 open.org/committees/xacml/docs/identifiers/">
539       <saml:AttributeDesignator AttributeName="physicianName"
540 AttributeNamespace="//medico.com/record/primaryCarePhysician/">
541     </equal>
542   </condition>
543 </ruleStatement>
```

544 Notes:

545 The post condition applies only to its immediately preceding sibling rule. While it may seem more desirable
546 to have an element that explicitly encloses the post-condition with the rule to which it applies, this solution
547 would lead to the inclusion of unnecessary tags in XACML instances when no post-condition is present.

548 3.5.4. Rule 4

549 The following XACML ruleStatement instance expresses Rule 4.

```
550 <?xml version="1.0" encoding="UTF-8"?>
551 <ruleStatement ruleId="//medico.com/rules/rule4"
552 metaPolicyRef="//www.oasis-open.org/committees/xacml/docs/meta-
553 policies/meta-policy-1" xmlns="http://www.oasis-
554 open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"
555 xmlns:saml="http://www.oasis-
556 open.org/committees/security/docs/draft-sstc-schema-assertion-
557 24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
558 xsi:schemaLocation="http://www.oasis-
559 open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd
560 D:\MYDOCU~1\Standards\XACML\V10SCH~1\XACMLV~4.XSD">
561 <comment>An administrator may read and write any record during
562 office hours and from a designated IP address sub-space</comment>
563 <target>
564 <subjects>
565 <saml:Attribute AttributeName="role"
566 AttributeNamespace="//medico.com">
567 <saml:AttributeValue>
568 <role
569 xmlns="http://www.w3.org/2001/XMLSchema/string">administrator</role>
570 </saml:AttributeValue>
571 </saml:Attribute>
572 </subjects>
573 <resources>
574 <saml:Attribute AttributeName="documentURI"
575 AttributeNamespace="//medico.com">
576 <saml:AttributeValue>
577 <documentURI
578 xmlns="http://www.w3.org/2001/XMLSchema/string">//medico.com/records
579 .*</documentURI>
580 </saml:AttributeValue>
581 </saml:Attribute>
582 </resources>
583 <actions>
584 <saml:Actions>
585 <saml:Action>read write</saml:Action>
586 </saml:Actions>
587 </actions>
588 </target>
589 <effect>permitIf</effect>
590 <condition>
591 <and>
592 <greaterOrEqual>
593 <saml:AttributeDesignator AttributeName="timeOfDay"
594 AttributeNamespace="//oasis-
595 open.org/committees/xacml/docs/identifiers/" />
596 <saml:Attribute AttributeName="time"
597 AttributeNamespace="http://www.w3.org/2001/XMLSchema/" />
598 <saml:AttributeValue>
599 <timeOfDay>08:00</timeOfDay>
600 </saml:AttributeValue>
601 </saml:Attribute>
602 </greaterOrEqual>
603 <lessOrEqual>
```

```
604         <saml:AttributeDesignator AttributeName="timeOfDay"
605 AttributeNamespace="//oasis-
606 open.org/committees/xacml/docs/identifiers/" />
607         <saml:Attribute AttributeName="time"
608 AttributeNamespace="http://www.w3.org/2001/XMLSchema/">
609         <saml:AttributeValue>
610         <timeOfDay>17:00</timeOfDay>
611         </saml:AttributeValue>
612         </saml:Attribute>
613     </lessOrEqual>
614     <patternMatch>
615     <saml:AttributeDesignator
616 AttributeName="authenticationLocality" AttributeNamespace="//oasis-
617 open.org/committees/xacml/docs/identifiers/" />
618     <saml:Attribute AttributeName="authenticationLocality"
619 AttributeNamespace="//medico.com">
620     <saml:AttributeValue>
621     <authenticationLocality
622 xmlns="http://www.w3.org/2001/XMLSchema/string">217.*</authenticatio
623 nLocality>
624     </saml:AttributeValue>
625     </saml:Attribute>
626     </patternMatch>
627     </and>
628 </condition>
629 </ruleStatement>
```

630 Notes:

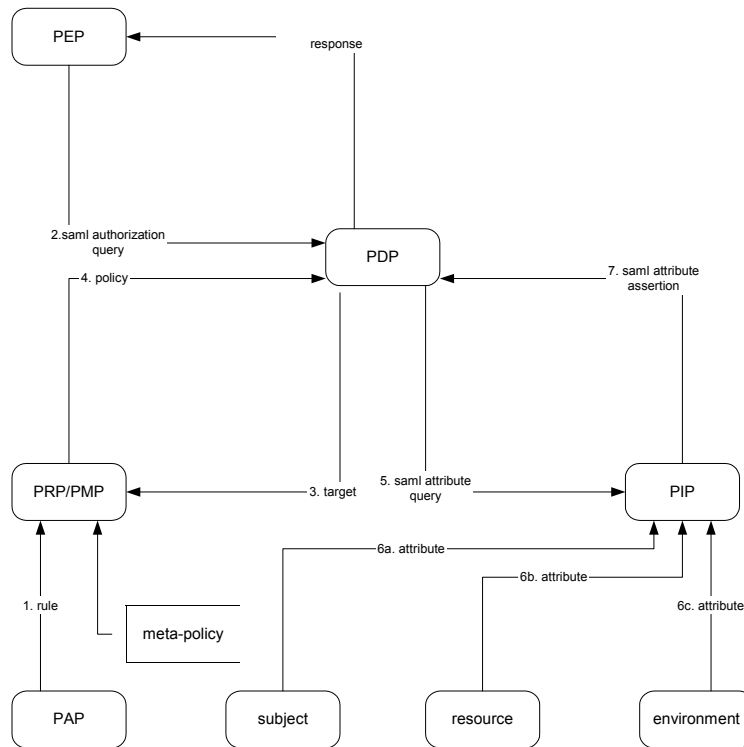
631 Pattern match uses the regular expression syntax.

632 **4. Models (non-normative)**

633 The context and schema of XACML are described in two models that describe different aspects of its
634 operation. These models are: the data-flow model and the policy language model. They are described in the
635 following sub-sections.

636 **4.1. Data-flow model**

637 The major actors in the XACML domain are shown in the data-flow diagram of Figure 1.



638

639

Figure 1 - Data-flow diagram

640 Some of the data-flows shown in the diagram may be facilitated by a repository. For instance, the
 641 communications between the PDP and the PIP may be facilitated by a repository, or the communications
 642 between the PDP and the PRP may be facilitated by a repository or the communication between the PAP and
 643 the PRP may be facilitated by a repository. The XACML specification is not intended to place restrictions on
 644 the location of any such repository, or indeed to prescribe a particular communication protocol for any of the
 645 data-flows.

646 The model operates by the following steps.

- 647 1. PAPs write rules and make them available to the PRP. From the point of view of an individual PAP, its
 648 rule statements may represent the complete policy for a particular target. However, the PRP may be
 649 aware of other PAPs that it considers authoritative for the same target. In which case, it is the PRP's job
 650 to obtain all the rules and (if necessary) use a PMP to remove any conflict amongst those rules and
 651 combine the rules in accordance with a meta-policy. The result should be a self-consistent rule
 652 statement.
- 653 2. The PEP sends an authorization decision request to the PDP, in the form of a SAML [SAML] request.
 654 The decision request contains some or all of the attributes required by the PDP to render a decision, in
 655 accordance with policy.
- 656 3. The PDP locates and retrieves the policy statement applicable to the decision request from the PRP.

- 657 4. The PRP returns the complete policy to the PDP in the form of an XACML rule statement or policy
658 instance. The PDP ensures that the decision request is in the scope of the policy statement or rule
659 statement.
- 660 5. The PDP examines the authorization decision request and the policy statement to ascertain whether it has
661 all the attribute values required to render an authorization decision. If it does not, then it requests
662 attributes from suitable PIPs in the form of SAML requests of the attribute query type [SAML].
- 663 6. The PIP (which may be a SAML attribute authority) may locate and retrieve the requested attributes from
664 other systems by a means, and in a form, that is out of scope for this specification.
- 665 7. The PIP returns the requested attributes to the PDP in the form of SAML responses containing SAML
666 attribute assertions.
- 667 8. The PDP evaluates the policy statement.
- 668 9. If the policy statement were to evaluate to TRUE, then the PDP returns an authorization decision, in the
669 form of a SAML response, to the PEP containing the "permit" decision attribute.

670 **4.2. Rule combination**

671 From the PRP's point of view, more than one rule may apply to a given target. For instance, taking the
672 example rules described in Section 3, the "date of birth" element is governed by rules 1,2,3 and 4. Therefore,
673 in plain language, the read rule applicable to "date of birth" is that at least one of the following conditions
674 must hold:

- 675 • The requestor is the patient;
- 676 • The requestor is the parent or guardian and the patient is under 16;
- 677 • The requestor is the primary care physician and a notification is sent to the patient; or
- 678 • The requestor is an administrator working from the office inside office hours.

679 XACML provides two alternative representations for combined rules. The first is a policy statement, which
680 is a set of component rules and an identifier for the meta-policy, defining how the component rules are to be
681 combined. The second is simply a rule statement containing the component rules explicitly combined as
682 defined by the meta-policy.

683 **4.2.1. Policy Statement**

684 The combined rules may be formed by referencing the individual rules in a policy statement.

```
685 <?xml version="1.0" encoding="UTF-8"?>  
686 <policyStatement policyId="//medico.com/policies/policy7"  
687 metaPolicyRef="//www.oasis-open.org/committees/xacml/docs/meta-  
688 policies/meta-policy-1" xmlns="http://www.oasis-  
689 open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"  
690 xmlns:saml="http://www.oasis-  
691 open.org/committees/security/docs/draft-sstc-schema-assertion-  
692 24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
693 xsi:schemaLocation="http://www.oasis-  
694 open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd  
695 D:\MYDOCU~1\Standards\XACML\V10SCH~1\XACMLV~4.XSD">  
696 <comment>The set of rules that govern access to a patient date of  
697 birth</comment>  
698 <target>  
699 <subjects>
```

```

700     <saml:Attribute AttributeName="RFC822Name"
701 AttributeNamespace="//medico.com">
702     <saml:AttributeValue>
703     <rfc822Name
704 xmlns="http://www.w3.org/2001/XMLSchema/string">*</rfc822Name>
705     </saml:AttributeValue>
706     </saml:Attribute>
707   </subjects>
708   <resources>
709     <saml:Attribute AttributeName="documentURI"
710 AttributeNamespace="//medico.com">
711     <saml:AttributeValue>
712     <documentURI
713 xmlns="http://www.w3.org/2001/XMLSchema/string">//medico.com/records
714 /patient/patientDoB</documentURI>
715     </saml:AttributeValue>
716     </saml:Attribute>
717   </resources>
718   <actions>
719     <saml:Actions>
720     <saml:Action>read</saml:Action>
721     </saml:Actions>
722   </actions>
723 </target>
724 <rule ruleID="//medico.com/rules/rule1"
725 authority="//medico.com"/>
726 <rule ruleID="//medico.com/rules/rule2"
727 authority="//medico.com"/>
728 <rule ruleID="//medico.com/rules/rule3"
729 authority="//medico.com"/>
730 <rule ruleID="//medico.com/rules/rule4"
731 authority="//medico.com"/>
732 </policyStatement>

```

733 4.2.2. Rule statement

734 Alternatively, the policy may be formed by creating a new rule statement that includes all the individual rule
735 conditions, combined according to the meta-policy.

```

736 <?xml version="1.0" encoding="UTF-8"?>
737 <policyStatement policyId="//medico.com/policies/policy1"
738 metaPolicyRef="//www.oasis-open.org/committees/xacml/docs/meta-
739 policies/meta-policy-1" xmlns="http://www.oasis-
740 open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"
741 xmlns:saml="http://www.oasis-
742 open.org/committees/security/docs/draft-sstc-schema-assertion-
743 24.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
744 xsi:schemaLocation="http://www.oasis-
745 open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd
746 D:\MYDOCU~1\Standards\XACML\V10SCH~1\XACMLV~1.XSD">
747   <comment>The set of policies that govern access to a patient date
748 of birth</comment>
749   <target>
750     <subjects>
751       <saml:Attribute AttributeName="RFC822Name"
752 AttributeNamespace="//medico.com">
753       <saml:AttributeValue>
754       <rfc822Name
755 xmlns="http://www.w3.org/2001/XMLSchema/string">*</rfc822Name>
756       </saml:AttributeValue>
757       </saml:Attribute>
758     </subjects>

```

```

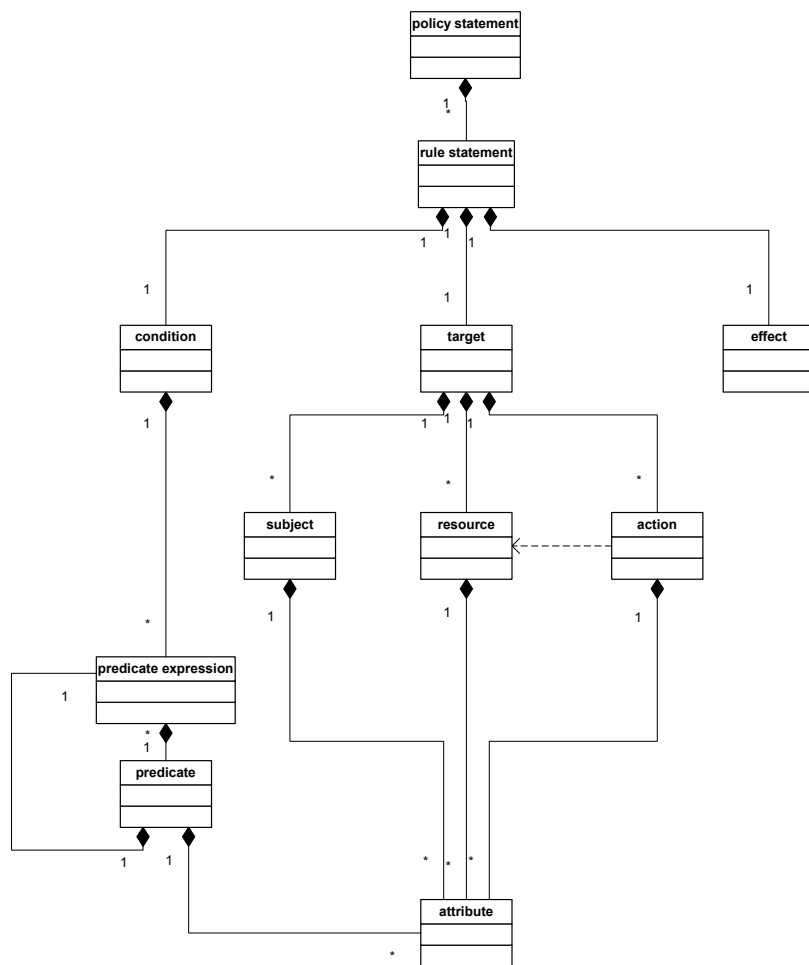
759     <resources>
760         <saml:Attribute AttributeName="documentURI"
761 AttributeNamespace="//medico.com">
762             <saml:AttributeValue>
763                 <documentURI
764 xmlns="http://www.w3.org/2001/XMLSchema/string">//medico.com/records
765 /patient/patientDoB</documentURI>
766             </saml:AttributeValue>
767         </saml:Attribute>
768     </resources>
769     <actions>
770         <saml:Actions>
771             <saml:Action>read</saml:Action>
772         </saml:Actions>
773     </actions>
774 </target>
775 <effect>permitIf</effect>
776 <condition>
777     <or>
778         <equal>
779             <saml:AttributeDesignator AttributeName="requestor"
780 AttributeNamespace="//oasis-
781 open.org/committees/xacml/docs/identifiers/">
782             <saml:AttributeDesignator AttributeName="patientName"
783 AttributeNamespace="//medico.com/record/patient/">
784         </equal>
785         <and>
786             <equal>
787                 <saml:AttributeDesignator AttributeName="requestor"
788 AttributeNamespace="//oasis-
789 open.org/committees/xacml/docs/identifiers/">
790                 <saml:AttributeDesignator
791 AttributeName="parentGuardianName"
792 AttributeNamespace="//medico.com/record/parentGuardian/">
793             </equal>
794             <greaterOrEqual>
795                 <saml:AttributeDesignator
796 AttributeName="patientDoB"
797 AttributeNamespace="//medico.com/record/patient/">
798                 <saml:Attribute AttributeName="dateOfBirth"
799 AttributeNamespace="//medico.com">
800                     <saml:AttributeValue>
801                         <dateOfBirth>1986-03-08</dateOfBirth>
802                     </saml:AttributeValue>
803                 </saml:Attribute>
804             </greaterOrEqual>
805         </and>
806     </equal>
807         <saml:AttributeDesignator AttributeName="requestor"
808 AttributeNamespace="//oasis-
809 open.org/committees/xacml/docs/identifiers/">
810             <saml:AttributeDesignator
811 AttributeName="physicianName"
812 AttributeNamespace="//medico.com/record/primaryCarePhysician/">
813         </equal>
814         <and>
815             <greaterOrEqual>
816                 <saml:AttributeDesignator AttributeName="timeOfDay"
817 AttributeNamespace="//oasis-
818 open.org/committees/xacml/docs/identifiers/">
819                 <saml:Attribute AttributeName="time"
820 AttributeNamespace="http://www.w3.org/2001/XMLSchema/">
821                     <saml:AttributeValue>

```

```
822         <timeOfDay>08:00</timeOfDay>
823         </saml:AttributeValue>
824     </saml:Attribute>
825 </greaterOrEqual>
826 <lessOrEqual>
827     <saml:AttributeDesignator AttributeName="timeOfDay"
828 AttributeNamespace="//oasis-
829 open.org/committees/xacml/docs/identifiers/">
830     <saml:Attribute AttributeName="time"
831 AttributeNamespace="http://www.w3.org/2001/XMLSchema/">
832     <saml:AttributeValue>
833         <timeOfDay>17:00</timeOfDay>
834     </saml:AttributeValue>
835     </saml:Attribute>
836 </lessOrEqual>
837 <patternMatch>
838     <saml:AttributeDesignator
839 AttributeName="authenticationLocality" AttributeNamespace="//oasis-
840 open.org/committees/xacml/docs/identifiers/">
841     <saml:Attribute
842 AttributeName="authenticationLocality"
843 AttributeNamespace="//medico.com">
844     <saml:AttributeValue>
845         <authenticationLocality
846 xmlns="http://www.w3.org/2001/XMLSchema/string">217.*</authenticatio
847 nLocality>
848     </saml:AttributeValue>
849     </saml:Attribute>
850 </patternMatch>
851 </and>
852 </or>
853 </condition>
854 </policyStatement>
```

855 **4.3. Policy language model**

856 The policy language model is shown in Figure 2.



857

858

Figure 2 - Policy language model

859 A rule statement contains:

- 860 • a ruleId;
- 861 • a target;
- 862 • an effect;
- 863 • a metaPolicyRef and
- 864 • a condition.

865 Target defines the set of names for the:

- 866 • resources;
- 867 • subjects and

868 • actions

869 to which the rule statement applies. If the rule statement applies to all entities of a particular type, then the
870 target definition is the root of the applicable name space. An XACML PDP MUST verify that the resources,
871 subjects and actions identified in an authorization decision request are each included in the target of the rule
872 statement that it uses to evaluate the request.

873 MetaPolicyRef specifies the meta-policy by which the rule statement MAY be combined with other rule
874 statements.

875 Condition is a general expression of predicates of attributes. It SHOULD NOT duplicate the exact predicates
876 implied by target. Therefore, it MAY be null.

877 Rule statements may be combined provided that their metaPolicyRefs are identical and that their targets are in
878 identical name spaces. This SHALL be done by including the conditions from the component rule statements
879 in the condition of the combined rule statement, in accordance with the meta-policy. (Section 7 for an
880 example of a meta-policy).

881 The target of the combined rule statement can be calculated from the targets of the component rule
882 statements. Two approaches are permitted:

- 883 • the target of the combined rule statement MAY contain the unions of the target definitions
884 for resource, subject and action that are contained in the component rule statements; or
- 885 • the target of the combined rule statement MAY contain the intersections of the definitions.

886 In the former case, the targets from the source rule statements MUST be included in the form of conditions in
887 the combined rule statement. In the latter case, this step MAY be omitted. In the former case, inapplicable
888 predicates should not be encountered. In the latter case, inapplicable predicates may be encountered.

889 The effect of the combined rule statement is defined by the meta-policy.

890 The syntax of a combined rule statement is identical to that of a component rule statement.

891 There is an alternative representation for a combined rules, called a policy statement. It contains:

- 892 • a policyId;
- 893 • a target;
- 894 • a metaPolicyRef and
- 895 • a set of rule references.

896 A rule reference identifies a rule statement by meta-data, (e.g. ruleID).

897 A policy statement can be converted to a rule statement at any time, by applying the meta-policy identified in
898 the rule statements to the rules. The resulting rule statement MUST contain a target computed from the target
899 values in the actual versions of the component rule statements used in the conversion, whereas, the target in
900 the policy statement SHOULD be computed from the target values in the actual versions of the component
901 rule statements at the time the policy statement was prepared.

902 The significance of the two representations is that, in the first case, the combined rule statement is generated
903 at the time of preparing the representation, whereas, in the second case, the combined rule statement can be
904 generated (using the latest versions of the component rule statements) immediately prior to evaluation. PDPs
905 SHOULD support both representations.

906

4.3.1. Target

907 The target of a rule statement identifies the set of authorization decision requests that the rule statement is
908 intended to evaluate. It contains definitions for resource, subject and action. In order for a rule statement to
909 apply to a specific authorization decision request, the resource, subject and action identified in the request
910 MUST be contained in the resource, subject and action definitions in the target. Target definitions are
911 discrete, in order that they may be indexed by the PDP.

912 Targets conform to a data model. XACML does not specify the data model, but it MUST be agreed between
913 the PAP and the PDP. The data model MUST be semi-hierarchical. That is, it MUST have one or more
914 disjoint trees for resources and one or more for subjects. Actions are leaf nodes of the resource node to which
915 they apply. Each level in the tree is identified with an attribute type. A "path" is a list of attribute-type/value
916 pairs linking a node to the root. The form of a target is a set of paths, one or more for each tree in the data
917 model.

918 A node MAY have more than one target associated with it.

919 An authorization decision request also specifies a set of paths by (directly or indirectly) providing resource,
920 subject and action attribute values.

921 A rule statement is applicable to a request if and only if every path in the target is part of a path in the request.

922 A PAP MAY store a rule statement in a repository, whose data model is congruent with that of the target data
923 model, by storing it at the lowest node of every path in the target. In practice, the rule statements may be
924 referenced from these nodes rather than stored at them.

925 When rule statements are combined, the combined target must be computed, and the repository must be
926 updated. Rule statements that conform to different target data models MUST NOT be combined.

927 The combined target SHALL be computed by separately combining trees of the same type from each of the
928 original targets. The combination may be in the form of a union or an intersection.

929 A union combination retains all of the original paths. If, as the result, all possible paths containing a
930 particular node are retained, then the path may be truncated at that node.

931 An intersection combination retains a path from one target if and only if it includes a path from the other
932 target.

933 The combined rule statement SHOULD be stored at the lowest node of every retained path.

934 Some attributes have an internal tree structure (e.g. DNS names). A sub-tree of this structure should be
935 represented by a regular expression. When such an attribute defines a level in a target tree, the sub-tree
936 defined by each node at that level SHALL be attached at that node.

937 **5. Policy syntax (normative, with the exception of the schema** 938 **fragments)**

939 **5.1. Element <policyStatement>**

940 The <policyStatement> element is a top-level element in the XACML schema.

941 `<xs:element name="policyStatement" type="xacml:PolicyStatementType"/>`

942 **5.2. Element <ruleStatement>**

943 The <ruleStatement> element is a top-level element in the XACML schema.

```
944 <xs:element name="ruleStatement" type="xacml:RuleStatementType"/>
```

945 **5.3. Complex type PolicyStatementType**

946 Elements of type PolicyStatementType extend the saml:StatementAbstractType so that they can be included
947 in a <saml:Assertion>. The <saml:Assertion> element contains some policy meta-data. The main elements
948 of this type definition are the <rule> and <target> elements and the metaPolicyRef attribute. The <rule>
949 element contains references to the set of rules that are to be combined in a policy. The <target> element is
950 computed from the target elements of the referenced <rule> elements either as an intersection or as a union.
951 And the metaPolicyRef attribute contains a reference to the meta-policy by which this <policyStatement>
952 element may be combined with other <policyStatement> elements.

```
953 <xs:complexType name="PolicyStatementType">  
954 <xs:complexContent>  
955 <xs:extension base="saml:StatementAbstractType">  
956 <xs:sequence>  
957 <xs:element name="comment" type="xs:string" minOccurs="0"/>  
958 <xs:element name="target" type="xacml:TargetType"/>  
959 <xs:element name="rule" type="xacml:RuleRefType"  
960 minOccurs="unbounded"/>  
961 </xs:sequence>  
962 <xs:attribute name="policyId" type="xs:anyURI" use="required"/>  
963 <xs:attribute name="policyName" type="xs:string" use="optional"/>  
964 <xs:attribute name="metaPolicyRef" type="xs:anyURI" use="required"/>  
965 </xs:extension>  
966 </xs:complexContent>  
967 </xs:complexType>
```

968 **5.4. Complex type RuleStatementType**

969 Like the PolicyStatementType, the RuleStatementType extends the saml:StatementAbstractType. The main
970 elements of this type definition are <target>, <effect>, <condition> and the metaPolicyRef attribute. The
971 metaPolicyRef attribute identifies the meta-policy by which an element of this type may be combined with
972 other elements of the type.

```
973 <xs:complexType name="RuleStatementType">  
974 <xs:complexContent>  
975 <xs:extension base="saml:StatementAbstractType">  
976 <xs:sequence>  
977 <xs:element name="comment" type="xs:string" minOccurs="0"/>  
978 <xs:element name="target" type="xacml:TargetType"/>  
979 <xs:element name="effect" type="xacml:EffectType"/>  
980 <xs:element name="condition"  
981 type="xacml:PredicateExpressionType" minOccurs="0"/>  
982 </xs:sequence>  
983 <xs:attribute name="ruleId" type="xs:anyURI" use="required"/>  
984 <xs:attribute name="ruleName" type="xs:string" use="optional"/>  
985 <xs:attribute name="metaPolicyRef" type="xs:anyURI" use="required"/>  
986 </xs:extension>  
987 </xs:complexContent>  
988 </xs:complexType>
```

989 **5.5. Complex type TargetType**

990 Target identifies the set of samlp:Requests of type samlp:AuthorizationDecisionQuery that the
991 <ruleStatement> is intended to address. It contains definition for subject, resource and action. If the subject,

992 resource and action identified in the <samlp:Request> are contained within the definition, then the
993 <ruleStatement> MAY be used to evaluate the <samlp:Request>.

```
994 <xs:complexType name="TargetType">  
995 <xs:sequence>  
996 <xs:element name="subjects" type="xacml:SubjectsType"/>  
997 <xs:element name="resources" type="xacml:ResourcesType"/>  
998 <xs:element name="actions" type="xacml:ActionsType"/>  
999 </xs:sequence>  
1000 </xs:complexType>
```

1001 **5.6. Complex type SubjectType**

1002 Elements of type SubjectType identify a set of subjects by a set of <saml:Attribute> elements.

```
1003 <xs:complexType name="SubjectsType">  
1004 <xs:sequence maxOccurs="unbounded">  
1005 <xs:element ref="saml:Attribute"/>  
1006 </xs:sequence>  
1007 </xs:complexType>
```

1008 **5.7. Complex type ResourceType**

1009 Elements of type ResourceType identify a set of resources by a list of <saml:Attribute> elements.

```
1010 <xs:complexType name="ResourcesType">  
1011 <xs:sequence maxOccurs="unbounded">  
1012 <xs:element ref="saml:Attribute"/>  
1013 </xs:sequence>  
1014 </xs:complexType>
```

1015 **5.8. Complex type ActionType**

1016 Elements of type ActionType identify a set of actions by a list of <saml:Action> elements.

```
1017 <xs:complexType name="ActionsType">  
1018 <xs:sequence>  
1019 <xs:element ref="saml:Actions"/>  
1020 </xs:sequence>  
1021 </xs:complexType>
```

1022 **5.9. Simple type EffectType**

1023 EffectType is an enumeration of "permitIf", "permitOnlyIf" and "denyIf". The attribute indicates the effect of
1024 the <ruleStatement> should it evaluate to TRUE. The relationship between the SAML Decision attribute
1025 issued by the PDP, the effect of the <ruleStatement> and the value of the <ruleStatement> is determined by
1026 meta-policy.

```
1027 <xs:simpleType name="EffectType">  
1028 <xs:restriction base="xs:string">  
1029 <xs:enumeration value="permitIf"/>  
1030 <xs:enumeration value="permitOnlyIf"/>  
1031 <xs:enumeration value="denyIf"/>  
1032 </xs:restriction>  
1033 </xs:simpleType>
```

1034 **5.10. Complex type PredicateExpressionType**

1035 Elements of type PredicateExpressionType contain either a <predicateExpression> or <predicate> element.
1036 These elements are both of abstract type. So, an element of this type may only contain an element in one of
1037 their substitution groups.

```
1038 <xs:complexType name="PredicateExpressionType">
1039   <xs:choice>
1040     <xs:element ref="xacml:predicateExpression"/>
1041     <xs:element ref="xacml:predicate"/>
1042   </xs:choice>
1043 </xs:complexType>
```

1044 **5.11. Element <predicateExpression>**

1045 This is the base element for predicate expression elements, such as <and> and <or>.

```
1046 <xs:element name="predicateExpression"
1047 type="xacml:PredicateExpressionAbstractType" abstract="true"/>
```

1048 **5.12. Complex type PredicateExpressionAbstractType**

```
1049 <xs:complexType name="PredicateExpressionAbstractType"/>
```

1050 **5.13. Element <predicate>**

1051 This is the base element for predicate elements, such as <equal> and <greaterOREqual>.

```
1052 <xs:element name="predicate" type="xacml:PredicateAbstractType"
1053 abstract="true"/>
```

1054 **5.14. Complex type PredicateAbstractType**

1055 This is an XACML extensibility point. New predicates may be added in the substitution group of
1056 <predicate>.

```
1057 <xs:complexType name="PredicateAbstractType"/>
```

1058 **5.15. Element <and>**

1059 The <and> element is an element of type AndType.

```
1060 <xs:element name="and" type="xacml:AndType"
1061 substitutionGroup="xacml:predicateExpression"/>
```

1062 **5.16. Element <or>**

1063 The <or> element is an element of type OrType.

```
1064 <xs:element name="or" type="xacml:OrType"
1065 substitutionGroup="xacml:predicateExpression"/>
```

1066 **5.17. Element <orderedOr>**

1067 The <orderedOr> element is an element of type OrderedOrType.

```
1068 <xs:element name="orderedOr" type="xacml:OrderedOrType"
1069 substitutionGroup="xacml:predicateExpression"/>
```

1070 **5.18. Element <nOf>**

1071 The <nOf> element is an element of type NOFType.

```
1072 <xs:element name="nOf" type="xacml:NOFType"
1073 substitutionGroup="xacml:predicateExpression"/>
```

1074 **5.19. Element <not>**

1075 The <not> element is an element of type NotType.

```
1076 <xs:element name="not" type="xacml:NotType"  
1077 substitutionGroup="xacml:predicateExpression"/>
```

1078 **5.20. Complex type AndType**

1079 Elements of type AndType can be evaluated with range true, false and indeterminate. They contain any
1080 combination of <predicateExpression> elements and <predicate> elements. Elements of this type evaluate to
1081 true if and only if all the elements contained in them evaluate to true.

```
1082 <xs:complexType name="AndType">  
1083 <xs:sequence minOccurs="0" maxOccurs="unbounded">  
1084 <xs:choice>  
1085 <xs:element ref="xacml:predicateExpression"/>  
1086 <xs:element ref="xacml:predicate"/>  
1087 </xs:choice>  
1088 </xs:sequence>  
1089 </xs:complexType>
```

1090 **5.21. Complex type OrType**

1091 Elements of type OrType can be evaluated with range true, false and indeterminate. They contain any
1092 combination of <predicateExpression> elements and <predicate> elements. Elements of this type evaluate to
1093 true if one or more of the elements contained in them evaluate to true.

```
1094 <xs:complexType name="OrType">  
1095 <xs:sequence minOccurs="0" maxOccurs="unbounded">  
1096 <xs:choice>  
1097 <xs:element ref="xacml:predicateExpression"/>  
1098 <xs:element ref="xacml:predicate"/>  
1099 </xs:choice>  
1100 </xs:sequence>  
1101 </xs:complexType>
```

1102 **5.22. Complex type OrderedOrType**

1103 Elements of type OrderedOrType can be evaluated with range true, false and indeterminate. They contain
1104 any combination of <predicateExpression> elements and <predicate> elements. Elements of this type
1105 evaluate to true if one or more of the elements contained in them evaluate to true. The <predicateExpression>
1106 elements and <predicate> elements MUST be evaluated in the order in which they appear, and evaluation
1107 MAY halt once one has evaluated to true.

```
1108 <xs:complexType name="OrderedOrType">  
1109 <xs:sequence minOccurs="0" maxOccurs="unbounded">  
1110 <xs:choice>  
1111 <xs:element ref="xacml:predicateExpression"/>  
1112 <xs:element ref="xacml:predicate"/>  
1113 </xs:choice>  
1114 </xs:sequence>  
1115 </xs:complexType>
```

1116 **5.23. Complex type NoFType**

1117 Elements of type NoFType can be evaluated with range true, false and indeterminate. They contain any
1118 combination of <predicateExpression> elements and <predicate> elements and a quorum attribute. Elements
1119 of this type evaluate to true if a number of the elements contained in them equal to the value of quorum
1120 evaluate to true.

```

1121 <xs:complexType name="NotType">
1122   <xs:sequence minOccurs="0" maxOccurs="unbounded">
1123     <xs:choice>
1124       <xs:element ref="xacml:predicateExpression"/>
1125       <xs:element ref="xacml:predicate"/>
1126     </xs:choice>
1127   </xs:sequence>
1128   <xs:attribute name="quorum" type="xs:positiveInteger"/>
1129 </xs:complexType>

```

1130 **5.24. Complex type NotType**

1131 Elements of type NotType can be evaluated with range true, false and indeterminate. They contain a choice
1132 of a <predicateExpression> element or a <predicate> element. Elements of this type evaluate to true if the
1133 element contained in them evaluates to false.

```

1134 <xs:complexType name="NotType">
1135   <xs:choice>
1136     <xs:element ref="xacml:predicateExpression" minOccurs="0"/>
1137     <xs:element ref="xacml:predicate" minOccurs="0"/>
1138   </xs:choice>
1139 </xs:complexType>

```

1140 **5.25. Element <present>**

1141 The <present> element is in the substitution group of <predicate>.

```

1142 <xs:element name="present" type="xacml:PresentType"
1143 substitutionGroup="xacml:predicate"/>

```

1144 **5.26. Element <equal>**

1145 The <equal> element is in the substitution group of <predicate>. It is of type CompareType. It SHALL
1146 evaluate to true if and only if the attributes it contains are equal.

```

1147 <xs:element name="equal" type="xacml:CompareType"
1148 substitutionGroup="xacml:predicate"/>

```

1149 **5.27. Element <greaterOrEqual>**

1150 The <greaterOrEqual> element is in the substitution group of <predicate>. It is of type CompareType. It
1151 MUST evaluate to TRUE only if the attribute value referenced by the first element of the CompareType is
1152 greater than or equal to the attribute value referenced by the second element. The elements must be of the
1153 same type, which may be string, normalizedString, byte, unsignedByte, base64Binary, hexBinary, integer,
1154 positiveInteger, negativeInteger, nonNegativeInteger, nonPositiveInteger, int, unsignedInt, long,
1155 unsignedLong, short, unsignedShort, decimal, float, double, time, dateTime, duration, date, gMonth, gYear,
1156 gYearMonth, gDay, gMonthDay, Name or QName.

```

1157 <xs:element name="greaterOrEqual" type="xacml:CompareType"
1158 substitutionGroup="xacml:predicate"/>

```

1159 **5.28. Element <lessOrEqual>**

1160 The <lessOrEqual> element is in the substitution group of <predicate>. It is of type CompareType. It MUST
1161 evaluate to TRUE only if the attribute value referenced by the first element of the CompareType is less than
1162 or equal to the attribute value referenced by the second element. The elements must be of the same type,
1163 which may be string, normalizedString, byte, unsignedByte, base64Binary, hexBinary, integer,
1164 positiveInteger, negativeInteger, nonNegativeInteger, nonPositiveInteger, int, unsignedInt, long,
1165 unsignedLong, short, unsignedShort, decimal, float, double, time, dateTime, duration, date, gMonth, gYear,
1166 gYearMonth, gDay, gMonthDay, Name or QName.

1167 `<xs:element name="lessOrEqual" type="xacml:CompareType"`
1168 `substitutionGroup="xacml:predicate"/>`

1169 **5.29. Element <subset>**

1170 The <subset> element is in the substitution group of <predicate>. It is of type CompareType. It MUST
1171 evaluate to TRUE only if the value referenced by the first element of the CompareType is amongst the set of
1172 values referenced by the second element.

1173 `<xs:element name="subset" type="xacml:CompareType"`
1174 `substitutionGroup="xacml:predicate"/>`

1175 **5.30. Element <superset>**

1176 The <superset> element is in the substitution group of <predicate>. It is of type CompareType. It MUST
1177 evaluate to TRUE only if the value referenced by the first element of the CompareType contains the set of
1178 values referenced by the second element.

1179 `<xs:element name="supersetOf" type="xacml:CompareType"`
1180 `substitutionGroup="xacml:predicate"/>`

1181 **5.31. Element <patternMatch>**

1182 The <patternMatch> element is in the substitution group of <predicate>. It is of type CompareType. It
1183 MUST evaluate to TRUE only if the string referenced by the first element of the CompareType matches the
1184 pattern defined in the string referenced by the second element. The pattern must be defined as a regular
1185 expression.

1186 `<xs:element name="patternMatch" type="xacml:CompareType"`
1187 `substitutionGroup="xacml:predicate"/>`

1188 **5.32. Element <nonNullSetInterscetion>**

1189 The <nonNullSetIntersection> element is in the substitution group of <predicate>. It contains an element of
1190 type CompareType. It MUST evaluate to TRUE only if the set of values referenced by the two elements of
1191 the CompareType have at least one value in common.

1192 `<xs:element name="nonNullSetIntersection" type="xacml:CompareType"`
1193 `substitutionGroup="xacml:predicate"/>`

1194 **5.33. Complex type PresentType**

1195 Elements of type PresentType contain an element of type <saml:AttributeDesignator>. It MUST evaluate to
1196 TRUE only if the element obtained by resolving the <saml:AttributeDesignator> element exists.

1197 `<xs:complexType name="PresentType">`
1198 `<xs:complexContent>`
1199 `<xs:extension base="xacml:PredicateAbstractType">`
1200 `<xs:sequence>`
1201 `<xs:element ref="saml:AttributeDesignator"/>`
1202 `</xs:sequence>`
1203 `</xs:extension>`
1204 `</xs:complexContent>`
1205 `</xs:complexType>`

1206 **5.34. Complex type CompareType**

1207 Elements of the type CompareType contain a pair of elements. The first is a <saml:AttributeDesignator>
1208 element and the second is either a <saml:AttributeDesignator> or a <saml:Attribute> element. The two
1209 elements MUST be of identical type and include an xsi:type attribute.

```

1210 <xs:complexType name="CompareType">
1211   <xs:complexContent>
1212     <xs:extension base="xacml:PredicateAbstractType">
1213       <xs:sequence>
1214         <xs:element ref="saml:AttributeDesignator"/>
1215         <xs:choice>
1216           <xs:element ref="saml:AttributeDesignator"/>
1217           <xs:element ref="saml:Attribute"/>
1218         </xs:choice>
1219       </xs:sequence>
1220     </xs:extension>
1221   </xs:complexContent>
1222   <!-- XML operands in "set" operations MUST be of type xs:list -->
1223   <!-- XML operands in "inequality" operations MUST contain an xsi:type
1224 attribute for which
1225 XACML defines a comparison algorithm -->
1226 </xs:complexType>

```

1227 **5.35. Complex type RuleRefType**

1228 Elements of type RuleRefType contain attributes by which instances of <ruleStatement> may be referenced.

```

1229 <xs:complexType name="RuleRefType">
1230   <xs:attribute name="ruleID" type="xs:anyURI"/>
1231   <xs:attribute name="authority" type="xs:anyURI"/>
1232 </xs:complexType>

```

1233 **6. XACML identifiers (normative)**

1234 This section defines standard identifiers for commonly-used entities.

1235 **6.1. Requestor**

1236 **6.2. Time of day**

1237 **6.3. Authentication locality**

1238 **6.4. Meta-policy one**

1239 **7. Meta-policy one (non-normative)**

1240 "Meta-policy one" is a meta-policy for defining the combining of rules into a policy. Only <ruleStatements>
1241 that have the value `"/www.oasis-open.org/committees/xacml/docs/meta-policies/meta-policy-1"` for
1242 their metaPolicyRef attribute SHALL be combined using this meta-policy.

```

1243 <metaPolicyStatement metaPolicyRef="/www.oasis-
1244 open.org/committees/xacml/docs/meta-policies/meta-policy-1">
1245   <and>
1246     <or>
1247       <policyRef>
1248         <policyId>//saml:assertion/ruleStatement
1249           [@metaPolicyRef="metaPolicy1"][@effect="permitIf"] |
1250           [@effect="permitOnlyIf"]/condition
1251         </policyId>
1252       </policyRef>
1253     </or>
1254     <not>
1255     </or>

```

```
1256     <policyRef>
1257         <policyId>//saml:assertion/ruleStatement
1258             [@metaPolicyRef="metaPolicy1"][@effect="denyIf"]/condition
1259     </policyId>
1260 </policyRef>
1261 </or>
1262 </not>
1263 <and>
1264     <policyRef>
1265         <policyId>//saml:assertion/ruleStatement
1266             [@metaPolicyRef="metaPolicy1"][@effect="permitOnlyif"]/condition
1267     </policyId>
1268 </policyRef>
1269 </and>
1270 </and>
1271 <effect>permitIf<effect>
1272 </metaPolicyStatement>
```

1273 **8. Profiles (non-normative)**

1274 Information in this section is normative, but not mandatory to implement.

1275 **8.1. XACML**

1276 Describes subsets of XACML appropriate to general classes of problem

1277 **8.2. SAML**

1278 Describes the subset of SAML that is relevant to XACML

1279 We need to specify SAML status codes for situations specific to XACML, such as:

- 1280 • PDP has no policy for the requested target
- 1281 • PDP cannot retrieve the required attributes

1282 **8.3. XML Digital Signature**

1283 Describes how XACML instances shall be integrity-protected in the case where XML DSig is used. PAPs
1284 MAY sign applicable policy. When a PRP combines applicable policies, it MAY sign the resulting
1285 applicable policy.

1286 Issue: Should the identities and/or signatures of the PAPs be preserved in the composed policy?

1287 **8.4. LDAP**

1288 Describes an LDAP schema for the case where LDAP is used to distribute XACML

1289 **9. XACML extension points (non-normative)**

1290 Describes the points within the XACML model and schema where extensions can be added

1291 **10. Security and privacy (non-normative)**

1292 Vulnerabilities and safeguards

11. References

- 1293 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1294 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997
- 1296 [SAML] Security Assertion Markup Language available from <http://www.oasis-open.org/committees/security/#documents>
- 1298 [XMLSig] D. Eastlake et al., *XML-Signature Syntax and Processing*,
1299 <http://www.w3.org/TR/xmlsig-core/>, World Wide Web Consortium.
- 1300 [XMLSig-XSD] XML Signature Schema available from <http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd>.
- 1301

12. Schema (normative)

1302 This appendix contains the XACML schema definition.

```
1304 <?xml version="1.0" encoding="UTF-8"?>
1305 <xs:schema targetNamespace="http://www.oasis-
1306 open.org/committees/xacml/docs/draft-xacml-schema-policy-10.xsd"
1307 xmlns:xacml="http://www.oasis-open.org/committees/xacml/docs/draft-xacml-schema-
1308 policy-10.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
1309 xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-
1310 assertion-24.xsd" elementFormDefault="qualified"
1311 attributeFormDefault="unqualified">
1312   <xs:import namespace="http://www.oasis-
1313 open.org/committees/security/docs/draft-sstc-schema-assertion-24.xsd"
1314 schemaLocation="D:\My Documents\Standards\Xacml\v10 schema\draft-sstc-schema-
1315 assertion-24.xsd"/>
1316   <xs:element name="policyStatement" type="xacml:PolicyStatementType"/>
1317   <xs:element name="ruleStatement" type="xacml:RuleStatementType"/>
1318   <xs:complexType name="PolicyStatementType">
1319     <xs:complexContent>
1320       <xs:extension base="saml:StatementAbstractType">
1321         <xs:sequence>
1322           <xs:element name="comment" type="xs:string" minOccurs="0"/>
1323           <xs:element name="target" type="xacml:TargetType"/>
1324           <xs:element name="rule" type="xacml:RuleRefType"
1325 maxOccurs="unbounded"/>
1326         </xs:sequence>
1327         <xs:attribute name="policyId" type="xs:anyURI" use="required"/>
1328         <xs:attribute name="policyName" type="xs:string" use="optional"/>
1329         <xs:attribute name="metaPolicyRef" type="xs:anyURI" use="required"/>
1330       </xs:extension>
1331     </xs:complexContent>
1332   </xs:complexType>
1333   <xs:complexType name="RuleStatementType">
1334     <xs:complexContent>
1335       <xs:extension base="saml:StatementAbstractType">
1336         <xs:sequence>
1337           <xs:element name="comment" type="xs:string" minOccurs="0"/>
1338           <xs:element name="target" type="xacml:TargetType"/>
1339           <xs:element name="effect" type="xacml:EffectType"/>
1340           <xs:element name="condition"
1341 type="xacml:PredicateExpressionType" minOccurs="0"/>
1342         </xs:sequence>
1343         <xs:attribute name="ruleId" type="xs:anyURI" use="required"/>
1344         <xs:attribute name="ruleName" type="xs:string" use="optional"/>
1345         <xs:attribute name="metaPolicyRef" type="xs:anyURI" use="required"/>
1346       </xs:extension>
1347     </xs:complexContent>
```



```

1348 </xs:complexType>
1349 <xs:complexType name="TargetType">
1350   <xs:sequence>
1351     <xs:element name="subjects" type="xacml:SubjectsType"/>
1352     <xs:element name="resources" type="xacml:ResourcesType"/>
1353     <xs:element name="actions" type="xacml:ActionsType"/>
1354   </xs:sequence>
1355 </xs:complexType>
1356 <xs:complexType name="SubjectsType">
1357   <xs:sequence maxOccurs="unbounded">
1358     <xs:element ref="saml:Attribute"/>
1359   </xs:sequence>
1360 </xs:complexType>
1361 <xs:complexType name="ResourcesType">
1362   <xs:sequence maxOccurs="unbounded">
1363     <xs:element ref="saml:Attribute"/>
1364   </xs:sequence>
1365 </xs:complexType>
1366 <xs:complexType name="ActionsType">
1367   <xs:sequence>
1368     <xs:element ref="saml:Actions"/>
1369   </xs:sequence>
1370 </xs:complexType>
1371 <xs:simpleType name="EffectType">
1372   <xs:restriction base="xs:string">
1373     <xs:enumeration value="permitIf"/>
1374     <xs:enumeration value="permitOnlyIf"/>
1375     <xs:enumeration value="denyIf"/>
1376   </xs:restriction>
1377 </xs:simpleType>
1378 <xs:complexType name="PredicateExpressionType">
1379   <xs:choice>
1380     <xs:element ref="xacml:predicateExpression"/>
1381     <xs:element ref="xacml:predicate"/>
1382   </xs:choice>
1383 </xs:complexType>
1384 <xs:element name="predicateExpression"
1385 type="xacml:PredicateExpressionAbstractType" abstract="true"/>
1386 <xs:complexType name="PredicateExpressionAbstractType">
1387   <xs:element name="and" type="xacml:AndType"
1388 substitutionGroup="xacml:predicateExpression"/>
1389   <xs:element name="or" type="xacml:OrType"
1390 substitutionGroup="xacml:predicateExpression"/>
1391   <xs:element name="orderedOr" type="xacml:OrderedOrType"
1392 substitutionGroup="xacml:predicateExpression"/>
1393   <xs:element name="noOf" type="xacml:NoOfType"
1394 substitutionGroup="xacml:predicateExpression"/>
1395   <xs:element name="not" type="xacml:NotType"
1396 substitutionGroup="xacml:predicateExpression"/>
1397   <xs:complexType name="AndType">
1398     <xs:sequence minOccurs="0" maxOccurs="unbounded">
1399       <xs:choice>
1400         <xs:element ref="xacml:predicateExpression"/>
1401         <xs:element ref="xacml:predicate"/>
1402       </xs:choice>
1403     </xs:sequence>
1404   </xs:complexType>
1405   <xs:complexType name="OrType">
1406     <xs:sequence minOccurs="0" maxOccurs="unbounded">
1407       <xs:choice>
1408         <xs:element ref="xacml:predicateExpression"/>
1409         <xs:element ref="xacml:predicate"/>
1410       </xs:choice>

```

```

1411     </xs:sequence>
1412 </xs:complexType>
1413 <xs:complexType name="OrderedOrType">
1414   <xs:sequence minOccurs="0" maxOccurs="unbounded">
1415     <xs:choice>
1416       <xs:element ref="xacml:predicateExpression"/>
1417       <xs:element ref="xacml:predicate"/>
1418     </xs:choice>
1419   </xs:sequence>
1420 </xs:complexType>
1421 <xs:complexType name="NoFType">
1422   <xs:sequence minOccurs="0" maxOccurs="unbounded">
1423     <xs:choice>
1424       <xs:element ref="xacml:predicateExpression"/>
1425       <xs:element ref="xacml:predicate"/>
1426     </xs:choice>
1427   </xs:sequence>
1428   <xs:attribute name="quorum" type="xs:positiveInteger"/>
1429 </xs:complexType>
1430 <xs:complexType name="NotType">
1431   <xs:choice>
1432     <xs:element ref="xacml:predicateExpression" minOccurs="0"/>
1433     <xs:element ref="xacml:predicate" minOccurs="0"/>
1434   </xs:choice>
1435 </xs:complexType>
1436 <xs:element name="predicate" type="xacml:PredicateAbstractType"
1437 abstract="true"/>
1438 <!--This is an XACML extensibility point.  New predicates may be added in the
1439 substitution group of "predicate"-->
1440 <xs:complexType name="PredicateAbstractType"/>
1441 <xs:element name="present" type="xacml:PresentType"
1442 substitutionGroup="xacml:predicate"/>
1443 <xs:element name="equal" type="xacml:CompareType"
1444 substitutionGroup="xacml:predicate"/>
1445 <xs:element name="greaterOrEqual" type="xacml:CompareType"
1446 substitutionGroup="xacml:predicate"/>
1447 <xs:element name="lessOrEqual" type="xacml:CompareType"
1448 substitutionGroup="xacml:predicate"/>
1449 <xs:element name="subset" type="xacml:CompareType"
1450 substitutionGroup="xacml:predicate"/>
1451 <xs:element name="superset" type="xacml:CompareType"
1452 substitutionGroup="xacml:predicate"/>
1453 <xs:element name="patternMatch" type="xacml:CompareType"
1454 substitutionGroup="xacml:predicate"/>
1455 <xs:element name="nonNullSetIntersection" type="xacml:CompareType"
1456 substitutionGroup="xacml:predicate"/>
1457 <xs:complexType name="PresentType">
1458   <xs:complexContent>
1459     <xs:extension base="xacml:PredicateAbstractType">
1460       <xs:sequence>
1461         <xs:element ref="saml:AttributeDesignator"/>
1462       </xs:sequence>
1463     </xs:extension>
1464   </xs:complexContent>
1465 </xs:complexType>
1466 <xs:complexType name="CompareType">
1467   <xs:complexContent>
1468     <xs:extension base="xacml:PredicateAbstractType">
1469       <xs:sequence>
1470         <xs:element ref="saml:AttributeDesignator"/>
1471         <xs:choice>
1472           <xs:element ref="saml:AttributeDesignator"/>
1473           <xs:element ref="saml:Attribute"/>

```

```
1474         </xs:choice>
1475     </xs:sequence>
1476 </xs:extension>
1477 </xs:complexContent>
1478 <!-- XML operands in "set" operations MUST be of type xs:list -->
1479 <!-- XML operands in "inequality" operations MUST contain an xsi:type
1480 attribute for which
1481 XACML defines a comparison algorithm -->
1482 </xs:complexType>
1483 <xs:complexType name="RuleRefType">
1484     <xs:attribute name="ruleID" type="xs:anyURI"/>
1485     <xs:attribute name="authority" type="xs:anyURI"/>
1486 </xs:complexType>
1487 </xs:schema>
```

1488 13. Conformance (normative)

1489 Not the test cases themselves, but a description of how the test cases should be used. The test cases will be a
1490 set of files on the XACML Web site

1491 Conformance claims MAY be made by either one of two components in the XACML model:

- 1492 1. An implementation of a policy administration points that produces policy statements that conform with
1493 the XACML schema; and
- 1494 2. An implementation of a policy decision point that produces decisions in response to decision requests on
1495 the basis of XACML policy statements that conform with the XACML schema.

1496 In the current version of the specification, implementations of a policy retrieval point that produce policy
1497 statements that conform with the XACML schema by combining XACML applicable policies are treated in
1498 the same way as policy administration points, from the point of view of conformance.

1499 Policy administration points MAY claim conformance with the XACML specification provided merely that
1500 they produce schema-compliant policy statements.

1501 Policy decision points MAY claim conformance with the XACML specification provided that they correctly
1502 execute the XACML conformance test suite provided:

1503 <http://www.oasis-open.org/> ...

1504 XACML Test Suite

1505 The test suite comprises three directories:

- 1506 • Decision requests
- 1507 • Policies
- 1508 • Authentication and attribute assertions
- 1509 • Decision assertions

1510 The decision requests directory contains a set of text/xml/samlp files that are valid SAML authorization
1511 decision request messages.

1512 The policies directory contains precisely one XACML policy file whose target includes includes each of the
1513 decision requests.

1514 The assertions directory contains an unordered set of text/xml/saml files containing the attributes required to
1515 evaluates the policies in the policies directory.

1516 The decisions directory contains an unordered set of tect/xml/samlp files that are valid SAML authorization
1517 decision responses.

1518 A conformant XACML PDP implementation shall create a decision assertion in response to each and every
1519 decision request. The decision responses are linked to the corresponding decision requests by the request ID
1520 attribute.

1521 XACML implementations that target an application domain other than SAML may use a tool or process that
1522 is not an integral part of the implementation to convert between the SAML test vectors and its private data
1523 representation.

1524 Disclaimer: Implementors SHALL NOT consider the test cases provided in the XACML conformance test
1525 suite as providing 100% test coverage. OASIS does not represent that a conformant implementation will
1526 operate correctly in all respects nor that it is fit for its purpose.

1527 **Appendix A. Notices**

1528 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might
1529 be claimed to pertain to the implementation or use of the technology described in this document or the extent
1530 to which any license under such rights might or might not be available; neither does it represent that it has
1531 made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in
1532 OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for
1533 publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a
1534 general license or permission for the use of such proprietary rights by implementors or users of this
1535 specification, can be obtained from the OASIS Executive Director.

1536 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
1537 other proprietary rights which may cover technology that may be required to implement this specification.
1538 Please address the information to the OASIS Executive Director.

1539 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS] 2001. All
1540 Rights Reserved.

1541 This document and translations of it may be copied and furnished to others, and derivative works that
1542 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
1543 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
1544 this paragraph are included on all such copies and derivative works. However, this document itself may not
1545 be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed
1546 for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in
1547 the OASIS Intellectual Property Rights document must be followed, or as required to translate it into
1548 languages other than English.

1549 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or
1550 assigns.

1551 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1552 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1553 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY
1554 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1555 PARTICULAR PURPOSE.