



1

2

OASIS eXtensible Access Control Markup Language (XACML)

3

4 **Document identifier:** draft-xacml-v0.13

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

6 **Publication date:** 9 May 2002

7 **Maturity Level:** Committee Draft

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

9 **Editors:**

10 Simon Godik, Simon Godik

11 Tim Moses, Entrust

12 **Contributors:**

13 Anne Anderson, Sun Microsystems

14 Bill Parducci, Bill Parducci

15 Carlisle Adams, Entrust

16 Don Flinn, Hitachi

17 Ernesto Damiani, University of Milan

18 Hal Lockhart, Entegritiy

19 Ken Yagen, Crosslogix

20 Konstantin Besnozov, Hitachi

21 Michiharu Kudo, IBM, Japan

22 Pierangela Samarati, University of Milan

23 Polar Humenn, Syracuse University

24 Sekhar Vajjhala, Sun Microsystems

25	Table of contents	
26	OASIS EXTENSIBLE ACCESS CONTROL MARKUP LANGUAGE (XACML)	1
27	1. GLOSSARY (NON-NORMATIVE)	6
28	1.1. Preferred terms	6
29	1.2. Related terms	7
30	2. INTRODUCTION (NON-NORMATIVE)	7
31	2.1. Notation	7
32	2.2. Schema Organization and Namespaces	8
33	3. EXAMPLE (NON-NORMATIVE)	8
34	3.1. Introduction to the example	8
35	3.2. Example medical record instance	8
36	3.3. Example authorization decision request	10
37	3.4. Example plain-language rules	11
38	3.5. Example XACML rule instances	11
39	3.5.1. Rule 1	11
40	3.5.2. Rule 2	12
41	3.5.3. Rule 3	13
42	3.5.4. Rule 4	14
43	4. MODELS (NON-NORMATIVE)	14
44	4.1. Data-flow model	15
45	4.2. Policy language model	16
46	4.2.1. Rule	17
47	4.2.2. Policy statement	19
48	4.2.3. Policy set statement	22
49	5. POLICY SYNTAX (NORMATIVE, WITH THE EXCEPTION OF THE SCHEMA	
50	FRAGMENTS)	23
51	5.1. Element <policySetStatement>	23
52	5.2. Element <policyStatement>	23

53	5.3.	Element <rule>	23
54	5.4.	Element <authorizationDecisionStatement>	23
55	5.5.	Complex type PolicySetStatementType	23
56	5.6.	Complex type PolicyStatementType	24
57	5.7.	Complex type RuleType	24
58	5.8.	Complex type AuthorizationDecisionStatementType	25
59	5.9.	Complex type TargetType	25
60	5.10.	Complex type SubjectType	25
61	5.11.	Complex type ResourceType	25
62	5.12.	Complex type ActionType	25
63	5.13.	Simple type EffectType	26
64	5.14.	Complex type ObligationsType	26
65	5.15.	Complex type ObligationType	26
66	5.16.	Complex type PredicateExpressionType	26
67	5.17.	Element <predicateExpression>	26
68	5.18.	Complex type PredicateExpressionAbstractType	27
69	5.19.	Element <and>	27
70	5.20.	Element <or>	27
71	5.21.	Element <orderedOr>	27
72	5.22.	Element <nOf>	27
73	5.23.	Element <not>	27
74	5.24.	Complex type AndType	27
75	5.25.	Complex type OrType	27
76	5.26.	Complex type OrderedOrType	28
77	5.27.	Complex type NofType	28
78	5.28.	Complex type NotType	28
79	5.29.	Element <predicate>	28
80	5.30.	Complex type PredicateAbstractType	28

81	5.31.	Element <present>	29
82	5.32.	Element <equal>	29
83	5.33.	Element <greaterOrEqual>	29
84	5.34.	Element <lessOrEqual>	30
85	5.35.	Element <subset>	30
86	5.36.	Element <superset>	30
87	5.37.	Element <patternMatch>	30
88	5.38.	Element <nonNullSetInterscetion>	30
89	5.39.	Complex type PresentType	Error! Bookmark not defined.
90	5.40.	Complex type CompareType	31
91	5.41.	Element <attributeFunction>	31
92	5.42.	Complex type AttributeFunctionAbstractType	32
93	5.43.	Element <plus>	32
94	5.44.	Element <minus>	32
95	5.45.	Element <times>	33
96	5.46.	Element <divide>	33
97	5.47.	Complex type ArgumentListType	33
98	5.48.	Complex type PolicySetType	34
99	5.49.	Complex type PolicySetDesignatorType	34
100	5.50.	Complex type PolicyDesignatorType	34
101	5.51.	Complex type RuleSetType	34
102	5.52.	Complex type RuleDesignatorType	34
103	6.	XACML IDENTIFIERS (NORMATIVE)	35
104	6.1.	Requestor	35
105	6.2.	Time of day	35
106	6.3.	Attributes	35
107	6.3.1.	X.500 distinguished name	35
108	6.3.2.	Unix file-system path	35

109	6.3.3. Uniform resource identifier	35
110	6.4. Authentication locality	35
111	6.5. Deny-overrides rule-combining algorithm	35
112	6.6. Deny-overrides policy-combining algorithm	35
113	7. COMBINING ALGORITHMS (NORMATIVE)	36
114	7.1. Deny-overrides	36
115	8. PROFILES (NORMATIVE BUT NOT MANDATORY TO IMPLEMENT)	37
116	8.1. XACML	37
117	8.2. SAML	37
118	8.3. XML Digital Signature	37
119	8.4. LDAP	37
120	9. XACML EXTENSION POINTS (NON-NORMATIVE)	40
121	9.1. Substitution groups	41
122	9.2. URIs	41
123	10. SECURITY AND PRIVACY (NON-NORMATIVE)	41
124	11. REFERENCES	41
125	12. SCHEMA (NORMATIVE)	42
126	13. CONFORMANCE (NORMATIVE)	45
127	APPENDIX A. NOTICES	47
128		

129

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

131 **1.1. Preferred terms**

132 *Access* - Performing an *action*

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

134 *Action* - An operation on a *resource*

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

136 *Attribute* - Characteristic of a *subject, resource, action or environment* that may be referenced in a *predicate*

137 *Authorization decision* - The result of evaluating an *applicable policy*. A function that evaluates to "permit,
138 deny or indeterminate", and (optionally) a set of *obligations*

139 *Condition* - An expression of *predicates*

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

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

142 *Effect* - The intended consequence of a satisfied *condition* (either permit or deny)

143 *Environment* - The set of *attributes* that are independent of a particular *subject, resource or action*

144 *Information request* - The request by a *PDP* to a *PIP* for *attributes*

145 *Obligation* - An action specified in a *policy* or *policy set* that should be performed in conjunction with the
146 issuance of an *authorization decision*

147 *Policy* - A set of *rules* and an identifier for the *rule-combining algorithm*

148 *Policy administration point (PAP)* - The system entity that creates a *policy* or *policy set*

149 *Policy-combining algorithm* - The procedure for combining the *target, obligations* and *rules* from multiple
150 *policies*

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

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

155 *Policy information point (PIP)* - The system entity that acts as a source of *attribute* values

156 *Policy retrieval point (PRP)* - The system entity that locates and retrieves *applicable policy* for a particular
157 *decision request*

158 *Policy set* - A set of *policies* and other *policy sets*

159 **Predicate** - A statement about *attributes* whose truth can be evaluated
160 **Resource** - Data, service, or system component
161 **Rule** - A *target*, an *effect* and a set of *conditions*
162 **Rule-combining algorithm** - The procedure for combining the *target*, *effect* and *conditions* from multiple
163 *rules*
164 **Subject** - An actor whose *attributes* may be referenced by a *predicate*
165 **Target** - The set of *decision requests*, identified by definitions for *resource*, *subject* and *action*, that a *rule*,
166 *policy* or *policy set* is intended to evaluate
167 **Target mapping** - The process of confirming that a *rule*, *policy* or *policy set* is applicable to an authorization
168 *decision request*

169 1.2. Related terms

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

172 For instance, the term *attribute* is used in place of the terms: *group* and *role*.

173 In place of the terms: *privilege*, *permission*, *authorization*, *entitlement* and *right*, we use the term *rule*.

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

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

176 2. Introduction (non-normative)

177 This specification defines the syntax and semantics for XML-encoded access control data structures. The
178 XACML schema is an extension schema of SAML.

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

180 2.1. Notation

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

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

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

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

191 Listings of XACML schemas appear like this.

192
193

Example code listings appear like this.

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

- 197 • The prefix `saml` : stands for the SAML assertion namespace.
- 198 • The prefix `ds` : stands for the W3C XML Signature namespace.
- 199 • The prefix `xs` : stands for the W3C XML Schema namespace.

200 This specification uses the following typographical conventions in text: `<XACMLElement>`,
201 `<ns:ForeignElement>`, `Attribute`, **Datatype**, `OtherCode`.

202 **2.2. Schema Organization and Namespaces**

203 The XACML data structures are defined in a schema associated with the following XML namespace:

204 <http://www.oasis-open.org/committees/xacml/docs/draft-xacml-schema-policy-013.xsd>

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

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

209 <http://www.w3.org/2000/09/xmlsig#>

210 **3. Example (non-normative)**

211 This section contains an example use of XACML for illustrative purposes.

212 **3.1. Introduction to the example**

213 This section contains an example XML document, an example SAML authorization decision request and
214 example XACML rules. The XML document is a medical record. Four separate rules are defined.

215 **3.2. Example medical record instance**

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

```
218 <?xml version="1.0" encoding="UTF-8"?>  
219 <record xmlns="medico.com/records.xsd"  
220 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
221 xsi:schemaLocation="medico.com/records.xsd  
222 D:\MYDOCU~1\Standards\XACML\record.xsd">  
223   <patient>  
224     <patientName>  
225       <first>Bartholomew</first>  
226       <last>Simpson</last>  
227     </patientName>  
228     <patientContact>  
229       <street>27 Shelbyville Road</street>  
230       <city>Springfield</city>  
231       <state>MA</state>  
232       <zip>12345</zip>
```

233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299

```
<phone>555.123.4567</phone>
<fax/>
<email/>
</patientContact>
<patientDoB xsi:type="date">1992-03-21</patientDoB>
<patientGender xsi:type="string">male</patientGender>
<policyNumber xsi:type="string">555555</policyNumber>
</patient>
<parentGuardian>
  <parentGuardianName>
    <first>Homer</first>
    <last>Simpson</last>
  </parentGuardianName>
  <parentGuardianContact>
    <street>27 Shelbyville Road</street>
    <city>Springfield</city>
    <state>MA</state>
    <zip>12345</zip>
    <phone>555.123.4567</phone>
    <fax/>
    <email>homers@aol.com</email>
  </parentGuardianContact>
</parentGuardian>
<primaryCarePhysician>
  <physicianName>
    <first>Julius</first>
    <last>Hibbert</last>
  </physicianName>
  <physicianContact>
    <street>1 First St</street>
    <city>Springfield</city>
    <state>MA</state>
    <zip>12345</zip>
    <phone>555.123.9012</phone>
    <fax>555.123.9013</fax>
    <email/>
  </physicianContact>
  <registrationID>ABC123</registrationID>
</primaryCarePhysician>
<insurer>
  <name>Blue Cross</name>
  <street>1234 Main St</street>
  <city>Springfield</city>
  <state>MA</state>
  <zip>12345</zip>
  <phone>555.123.5678</phone>
  <fax>555.123.5679</fax>
  <email/>
</insurer>
<medical>
  <treatment>
    <drug>
      <name>methylphenidate hydrochloride</name>
      <dailyDosage>30mgs</dailyDosage>
      <startDate>1999-01-12</startDate>
    </drug>
    <comment>patient exhibits side-effects of skin coloration and
carpal degeneration</comment>
  </treatment>
  <result>
    <test>blood pressure</test>
    <value>120/80</value>
    <date>2001-06-09</date>
    <performedBy>Nurse Betty</performedBy>
  </result>
</medical>
</record>
```

300

3.3. Example authorization decision request

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

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

```
<?xml version="1.0" encoding="UTF-8"?>
<Request RequestID="47823081" MajorVersion="0" MinorVersion="28"
IssueInstant="2002-03-22T08:23:47-05:00" xmlns="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-protocol-28.xsd"
xmlns="http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-
protocol-28.xsd" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-sstc-
schema-assertion-28.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-protocol-28.xsd
D:\MYDOCU~1\Standards\XACML\V12SCH~1\draft-sstc-schema-protocol-28.xsd">
  <AuthorizationDecisionQuery
Resource="//medico.com/record/patient[@patientName/first='Bartholomew'][@pat
ientName/last='Simpson']/patientDoB">
    <saml:Subject>
      <saml:NameIdentifier NameQualifier="//medico.com">Julius
Hibbert</saml:NameIdentifier>
    </saml:Subject>
    <saml:Action>read</saml:Action>
    <saml:Evidence>
      <saml:Assertion AssertionID="64578390" Issuer="medico.com"
IssueInstant="2002-03-08T08:23:47-05:00" MajorVersion="0" MinorVersion="28"
xmlns="http://www.oasis-open.org/committees/security/docs/draft-sstc-schema-
assertion-28.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-28.xsd
D:\MYDOCU~1\Standards\XACML\V10SCH~1\draft-sstc-schema-assertion-28.xsd">
      <saml:AuthenticationStatement AuthenticationInstant="2002-03-
08T08:23:45-05:00" AuthenticationMethod="http://www.oasis-
open.org/committees/security/docs/draft-sstc-core-28/password-sha1">
        <saml:Subject>
          <saml:NameIdentifier
NameQualifier="//medico.com">Julius Hibbert</saml:NameIdentifier>
          <saml:SubjectConfirmation>
            <saml:ConfirmationMethod>http://www.oasis-
open.org/committees/security/docs/draft-sstc-core-
24/artifact</saml:ConfirmationMethod>
          </saml:SubjectConfirmation>
        </saml:Subject>
        <saml:AuthenticationLocality IPAddress="217.57.95.242"/>
      </saml:AuthenticationStatement>
    </saml:Assertion>
    <saml:Assertion MajorVersion="0" MinorVersion="28"
AssertionID="68938960" Issuer="medico.com" IssueInstant="2000-06-
15T15:02:39-05:00" xmlns="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-28.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-28.xsd
D:\MYDOCU~1\Standards\XACML\V10SCH~1\draft-sstc-schema-assertion-28.xsd">
      <saml:AttributeStatement>
        <saml:Subject>
          <saml:NameIdentifier
NameQualifier="//medico.com">Julius Hibbert</saml:NameIdentifier>
        </saml:Subject>
        <saml:Attribute AttributeName="role"
AttributeNamespace="//medico.com">
          <saml:AttributeValue>physician</saml:AttributeValue>
        </saml:Attribute>
      </saml:AttributeStatement>
    </saml:Assertion>
  </AuthorizationDecisionQuery>
</Request>
```

367
368
369
370

```
</saml:Assertion>  
</saml:Evidence>  
</AuthorizationDecisionQuery>  
</Request>
```

371 **3.4. Example plain-language rules**

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

- 373 1. A person may read any record for which he or she is the designated patient.
- 374 2. A person may read any record for which he or she is the designated parent or guardian, and for which the
375 patient is under 16 years of age.
- 376 3. A physician may write any medical element for which he or she is the designated primary care physician,
- 377 4. An administrator shall not be permitted to read or write any medical element.

378 These rules may be written by different *PAPs*, operating independently, or by a single *PAP*.

379 **3.5. Example XACML rule instances**

380 **3.5.1. Rule 1**

381 Rule 1 illustrates a simple rule with a single condition. The following XACML <rule> instance expresses
382 Rule 1.

383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418

```
<?xml version="1.0" encoding="UTF-8"?>  
<rule ruleId="//medico.com/rules/rule1" effect="Permit"  
xmlns="http://www.oasis-open.org/committees/xacml/docs/draft-xacml-schema-  
policy-12.xsd" xmlns:saml="http://www.oasis-  
open.org/committees/security/docs/draft-sstc-schema-assertion-28.xsd"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation="http://www.oasis-open.org/committees/xacml/docs/draft-  
xacml-schema-policy-12.xsd  
D:\MYDOCU~1\Standards\XACML\V12SCH~1\XACMLV~3.XSD">  
  <description>A person may read any record for which he or she is the  
  designated patient</description>  
  <target>  
    <subjects>  
      <saml:Attribute AttributeName="RFC822Name"  
AttributeNamespace="//medico.com">  
        <saml:AttributeValue>*</saml:AttributeValue>  
      </saml:Attribute>  
    </subjects>  
    <resources>  
      <saml:Attribute AttributeName="documentURI"  
AttributeNamespace="//medico.com">  
        <saml:AttributeValue//medico.com/record.*</saml:AttributeValue>  
      </saml:Attribute>  
    </resources>  
    <actions>  
      <saml:Action>read</saml:Action>  
    </actions>  
  </target>  
  <condition>  
    <equal>  
      <saml:AttributeDesignator AttributeName="requestor"  
AttributeNamespace="//oasis-open.org/committees/xacml/docs/identifiers/" />  
      <saml:AttributeDesignator AttributeName="patientName"  
AttributeNamespace="//medico.com/record/patient/" />  
    </equal>
```

419
420

```
</condition>  
</rule>
```

421

3.5.2. Rule 2

422
423

Rule 2 illustrates the use of a mathematical function, i.e. the <minus> function to calculate age. It also illustrates the use of predicate expressions, with the <and> and <not> elements.

424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478

```
<?xml version="1.0" encoding="UTF-8"?>  
<rule ruleId="//medico.com/rules/rule2" effect="Permit"  
xmlns="http://www.oasis-open.org/committees/xacml/docs/draft-xacml-schema-  
policy-12.xsd" xmlns:saml="http://www.oasis-  
open.org/committees/security/docs/draft-sstc-schema-assertion-28.xsd"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation="http://www.oasis-open.org/committees/xacml/docs/draft-  
xacml-schema-policy-12.xsd  
D:\MYDOCU~1\Standards\XACML\V12SCH~1\XACMLV~3.XSD">  
  <description>A person may read any record for which he or she is the  
  designated parent or guardian, and for which the patient is under 16 years  
  of age</description>  
  <target>  
    <subjects>  
      <saml:Attribute AttributeName="RFC822Name"  
AttributeNamespace="//medico.com">  
        <saml:AttributeValue>*/</saml:AttributeValue>  
      </saml:Attribute>  
    </subjects>  
    <resources>  
      <saml:Attribute AttributeName="documentURI"  
AttributeNamespace="//medico.com">  
        <saml:AttributeValue>//medico.com/record.*</saml:AttributeValue>  
      </saml:Attribute>  
    </resources>  
    <actions>  
      <saml:Action>read</saml:Action>  
    </actions>  
  </target>  
  <condition>  
    <and>  
      <equal>  
        <saml:AttributeDesignator AttributeName="requestor"  
AttributeNamespace="//oasis-open.org/committees/xacml/docs/identifiers/" />  
        <saml:AttributeDesignator AttributeName="parentGuardianName"  
AttributeNamespace="//medico.com/record/parentGuardian/" />  
      </equal>  
      <not>  
        <greaterOrEqual>  
          <minus>  
            <saml:AttributeDesignator AttributeName="today'sDate"  
AttributeNamespace="//medico.com/" />  
            <saml:AttributeDesignator AttributeName="patientDoB"  
AttributeNamespace="//medico.com/record/patient/" />  
          </minus>  
          <saml:Attribute AttributeName="ageOfConsent"  
AttributeNamespace="//medico.com">  
            <saml:AttributeValue>16-0-0</saml:AttributeValue>  
          </saml:Attribute>  
        </greaterOrEqual>  
      </not>  
    </and>  
  </condition>  
</rule>
```

479

3.5.3. Rule 3

480

Rule 3 illustrates the use of an *obligation*. The XACML <rule> element syntax does not include an element suitable for carrying an *obligation*, therefore Rule 3 has to be formatted as a <policyStatement> element, which is a type of SAML assertion.

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

```
<?xml version="1.0" encoding="UTF-8"?>
<saml:Assertion MajorVersion="0" MinorVersion="24" AssertionID="A7F34K90"
Issuer="medico.com" IssueInstant="2002-03-22T08:23:47-05:00">
  <policyStatement policyId="//medico.com/rules/policy3"
ruleCombiningAlgId="//www.oasis-
open.org/committees/xacml/docs/ruleCombiningAlgorithms/denyOverrides"
xmlns="http://www.oasis-open.org/committees/xacml/docs/draft-xacml-schema-
policy-12.xsd" xmlns:saml="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-28.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oasis-open.org/committees/xacml/docs/draft-
xacml-schema-policy-12.xsd
D:\MYDOCU~1\Standards\XACML\V12SCH~1\XACMLV~3.XSD">
  <description>A physician may write any medical element for which he
or she is the designated primary care physician, provided an email is sent
to the patient</description>
  <target>
    <subjects>
      <saml:Attribute AttributeName="role"
AttributeNamespace="//medico.com">
        <saml:AttributeValue>physician</saml:AttributeValue>
      </saml:Attribute>
    </subjects>
    <resources>
      <saml:Attribute AttributeName="documentURI"
AttributeNamespace="//medico.com">
        <saml:AttributeValue>//medico.com/record/medical.*</saml:AttributeValue>
      </saml:Attribute>
    </resources>
    <actions>
      <saml:Action>write</saml:Action>
    </actions>
  </target>
  <ruleSet>
    <rule ruleId="//medico.com/rules/rule3" effect="Permit">
      <description>A physician may write any medical element for
which he or she is the designated primary care physician</description>
      <target>
        <subjects>
          <saml:Attribute AttributeName="role"
AttributeNamespace="//medico.com">
            <saml:AttributeValue>physician</saml:AttributeValue>
          </saml:Attribute>
        </subjects>
        <resources>
          <saml:Attribute AttributeName="documentURI"
AttributeNamespace="//medico.com">
            <saml:AttributeValue>//medico.com/record/medical.*</saml:AttributeValue>
          </saml:Attribute>
        </resources>
        <actions>
          <saml:Action>write</saml:Action>
        </actions>
      </target>
      <condition>
        <and>
          <equal>
            <saml:AttributeDesignator
AttributeNamespace="//oasis-
open.org/committees/xacml/docs/identifiers/" />
```

```

546         <saml:AttributeDesignator
547 AttributeName="physicianName"
548 AttributeNamespace="//medico.com/record/primaryCarePhysician/" />
549         </equal>
550         <present>
551         <saml:AttributeDesignator
552 AttributeName="patient/email" AttributeNamespace="//medico.com/record/" />
553         </present>
554     </and>
555 </condition>
556 </rule>
557 </ruleSet>
558 <obligations>
559     <obligation ObligationId="//medico.com/emailer"
560 fulfilOn="Permit">
561         <saml:AttributeDesignator AttributeName="patient/email"
562 AttributeNamespace="//medico.com/record/" />
563         <saml:Attribute AttributeName="messageText"
564 AttributeNamespace="//medico.com/" />
565         <saml:AttributeValue>Your medical record has been accessed
566 by:</saml:AttributeValue>
567         </saml:Attribute>
568         <saml:AttributeDesignator AttributeName="requestor"
569 AttributeNamespace="//oasis-open.org/committees/xacml/docs/identifiers/" />
570     </obligation>
571 </obligations>
572 </policyStatement>
573 </saml:Assertion>

```

574 3.5.4. Rule 4

575 Rule 4 illustrates the use of the "deny" effect, and a rule with no <condition> element.

```

576 <?xml version="1.0" encoding="UTF-8"?>
577 <rule ruleId="//medico.com/rules/rule4" effect="Deny"
578 xmlns="http://www.oasis-open.org/committees/xacml/docs/draft-xacml-schema-
579 policy-12.xsd" xmlns:saml="http://www.oasis-
580 open.org/committees/security/docs/draft-sstc-schema-assertion-28.xsd"
581 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
582 xsi:schemaLocation="http://www.oasis-open.org/committees/xacml/docs/draft-
583 xacml-schema-policy-12.xsd
584 D:\MYDOCU~1\Standards\XACML\V12SCH~1\XACMLV~3.XSD">
585     <description>An administrator shall not be permitted to read or write
586 medical elements of a patient record</description>
587     <target>
588         <subjects>
589             <saml:Attribute AttributeName="role"
590 AttributeNamespace="//medico.com">
591                 <saml:AttributeValue>administrator</saml:AttributeValue>
592             </saml:Attribute>
593         </subjects>
594         <resources>
595             <saml:Attribute AttributeName="documentURI"
596 AttributeNamespace="//medico.com">
597                 <saml:AttributeValue>//medico.com/record/medical.*</saml:AttributeValue>
598             </saml:Attribute>
599         </resources>
600         <actions>
601             <saml:Action>read write</saml:Action>
602         </actions>
603     </target>
604 </rule>

```

606 4. Models (non-normative)

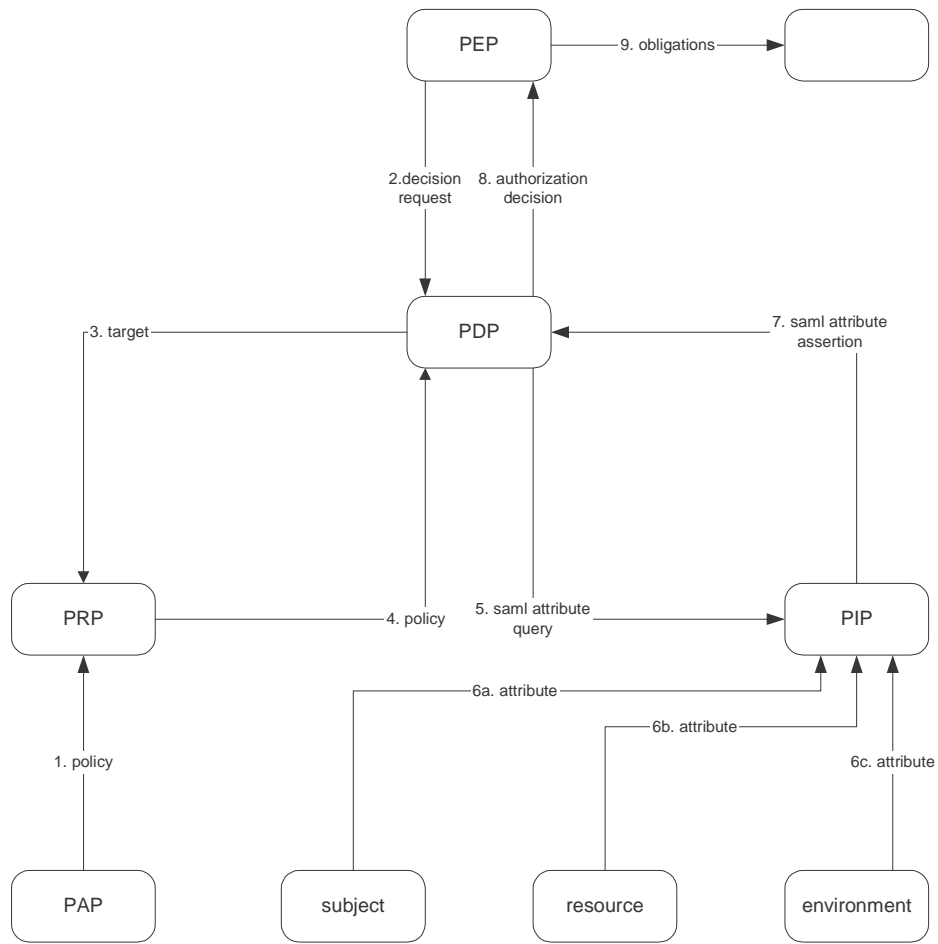
607 The context and schema of XACML are described in two models. These models are: the data-flow model
608 and the policy language model. They are described in the following sub-sections.

609

4.1. Data-flow model

610

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



611

612

Figure 1 - Data-flow diagram

613

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

618

The model operates by the following steps.

619

1. **PAPs** write **policies** and make them available to the **PRP**. From the point of view of an individual **PAP**, its **policies** represent the complete policy for a particular **target**. However, the **PDP** may be aware of other **PAPs** that it considers authoritative for the same **target**. In which case, it is the **PDP's** job to obtain all the **policies** and combine them in accordance with a **policy-combining algorithm**. The result should be a self-consistent **policy set**.

623

624

2. The **PEP** sends an authorization **decision request** to the **PDP**, in the form of a SAML [SAML] request. The **decision request** contains some or all of the **attributes** required by the **PDP** to render an **authorization decision**, in accordance with **applicable policy**.

626

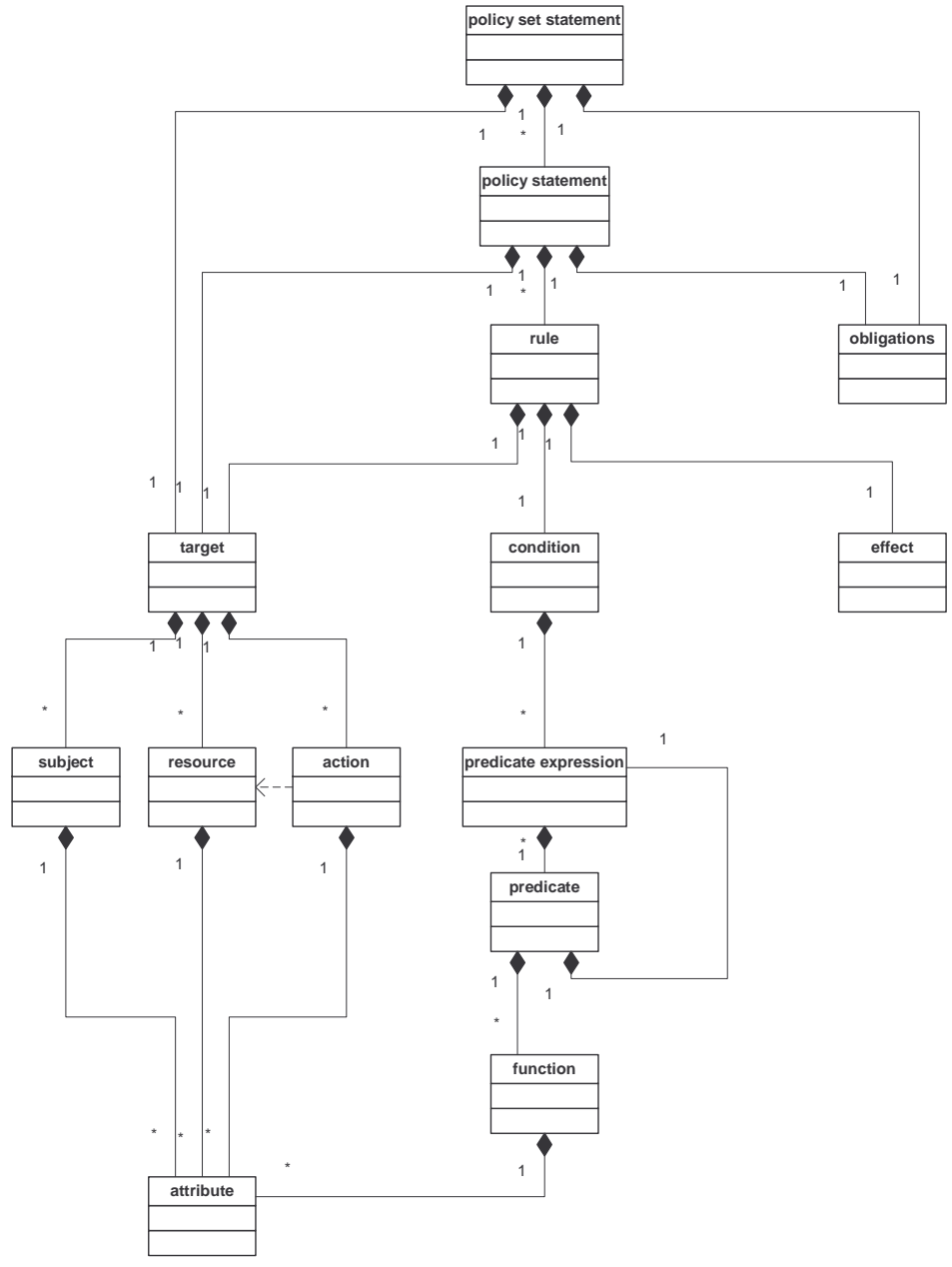
- 627 3. The *PDP* locates and retrieves the *policy* applicable to the *decision request* from the *PRP*.
- 628 4. The *PRP* returns the *applicable policy* to the *PDP* in the form of an XACML <policyStatement> or
629 <policySetStatement>. The *PDP* ensures that the *decision request* is in the scope of the
630 <policyStatement> or <policySetStatement>.
- 631 5. The *PDP* examines the authorization *decision request* and the *policy* to ascertain whether it has all the
632 *attribute* values required to render an *authorization decision*. If it does not, then it requests *attributes*
633 from suitable *PIPs* in the form of SAML requests of the attribute query type [SAML].
- 634 6. The *PIP* (which may be a SAML attribute authority) locates and retrieves the requested *attributes* from
635 other systems by a means, and in a form, that is out of scope for this specification.
- 636 7. The *PIP* returns the requested *attributes* to the *PDP* in the form of SAML responses containing SAML
637 attribute assertions. The *PDP* evaluates the *policy*.
- 638 8. If the *policy* were to evaluate to TRUE, then the *PDP* returns an *authorization decision*, in the form of a
639 SAML response, to the *PEP* containing the "Permit" `saml:Decision` attribute and (optional)
640 *obligations*.
- 641 9. The *PEP* fulfils the *obligations*.

642 **4.2. Policy language model**

643 The policy language model is shown in Figure 2. The main components of the model are:

- 644 • *Rule*;
- 645 • *Policy statement*; and
- 646 • *Policy set statement*.

647 These are described in the following sub-sections.



648

649

Figure 2 - Policy language model

650

4.2.1. Rule

651

The main components of a *rule* are:

652

- a *target*;

653

- an *effect*; and

654

- a *condition*.

655 These are discussed in the following sub-sections.

656 **4.2.1.1. Target**

657 The *target* defines the set of:

- 658 • *resources*;
- 659 • *subjects*; and
- 660 • *actions*

661 to which the *rule* is intended to apply. If the *rule* is intended to apply to all entities of a particular type, then
662 the *target* definition is the root of the applicable name space. An XACML *PDP* verifies that the *resources*,
663 *subjects* and *actions* identified in the authorization *decision request* are each included in the *target* of the
664 *rules* that it uses to evaluate the *decision request*. *Target* definitions are discrete, in order that they may be
665 indexed by the *PDP*.

666 **4.2.1.2. Effect**

667 The *effect* indicates the rule-writer's intended consequence of a true evaluation for the *rule*. Two values are
668 allowed: permit and deny.

669 **4.2.1.3. Condition**

670 *Condition* is a general expression of *predicates* of *attributes*. It should not duplicate the exact *predicates*
671 implied by the *target*. Therefore, it may be null.

672 **4.2.1.4. Rule evaluation**

673 A *rule* has a value that can be calculated by evaluating its contents. *Rule* evaluation involves separate
674 evaluation of the *rule's target* and *condition*. The *rule* truth table is shown in Table 1.

Target	Condition	Rule
Match	True	Effect
Match	False	Not applicable
Match	Indeterminate	Indeterminate
No-match	True	Not applicable
No-match	False	Not applicable
No-match	Indeterminate	Not applicable

675 **Table 1 - Rule truth table**

676 The *target* value is Match if the *resource*, *subject* and *action* specified in the *decision request* are each in the
677 *target* defined in the *rule*. Otherwise, its value is No-match.

678 The *condition* value is True if the <condition> element is null, or if it evaluates True for the *attribute* values
679 supplied in, or referenced by, the *decision request*. Its value is False if the <condition> element evaluates
680 False for the *attribute* values supplied in, or referenced by, the *decision request*. If any *attribute* value
681 referenced in the *condition* cannot be obtained, then the *condition* evaluates Indeterminate.

682 4.2.2. Policy statement

683 From the data-flow model one can see that *rules* are not exchanged amongst system entities. Therefore, a
684 *PAP* combines *rules* in a *policy*. A *policy* comprises four main components:

- 685 • a *target*;
- 686 • a *rule-combining algorithm*-identifier;
- 687 • a set of *rules*; and
- 688 • *obligations*.

689 4.2.2.1. Target

690 The *target* of a *policy* must include all the *decision requests* that it is intended to evaluate. The *target* may be
691 declared by the writer of the *policy*, or computed from the *targets* of its component *rules*.

692 If the *target* of the *policy statement* is computed from the *targets* of the component *rules*, two approaches are
693 permitted:

- 694 • the *target* of the *policy* may be the union of the *target* definitions for *resource*, *subject* and *action* that
695 are contained in the component *rules*; or
- 696 • the *target* of the *policy* may be the intersection of the *target* definitions for *resource*, *subject* and *action*
697 that are contained in the component *rules*.

698 In the former case, the *target* may be omitted from the individual *rules*, and the *targets* from the component
699 *rules* must be included in the form of *conditions* in their respective *rules*. As an example, the following *rule*
700 *target* and *condition* may be merged in a single *condition*.

```
701 <target>  
702   <subjects>  
703     <saml:Attribute AttributeName="RFC822Name"  
704     AttributeNamespace="//medico.com">  
705       <saml:AttributeValue>*/saml:AttributeValue>  
706     </saml:Attribute>  
707   </subjects>  
708   <resources>  
709     <saml:Attribute AttributeName="documentURI"  
710     AttributeNamespace="//medico.com">  
711       <saml:AttributeValue>//medico.com/records.*</saml:AttributeValue>  
712     </saml:Attribute>  
713   </resources>  
714   <actions>  
715     <saml:Action>read</saml:Action>  
716   </actions>  
717 </target>
```

718

```
719 <condition>  
720 <equal>
```

721
722
723
724
725
726

```
<saml:AttributeDesignator AttributeName="requestor"
AttributeNamespace="//oasis-open.org/committees/xacml/docs/identifiers/" />
<saml:AttributeDesignator AttributeName="patientName"
AttributeNamespace="//medico.com/record/patient/" />
</equal>
</condition>
```

727 Following is the merged *condition*.

728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756

```
<condition>
<and>
<equal>
<saml:AttributeDesignator AttributeName="documentURI"
AttributeNamespace="//medico.com/" />
<saml:Attribute AttributeName="documentURI"
AttributeNamespace="//medico.com">
<saml:AttributeValue>//medico.com/records.*</saml:AttributeValue>
</saml:Attribute>
</equal>
<equal>
<saml:AttributeDesignator AttributeName="Actions"
AttributeNamespace="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-24" />
</saml:Attribute AttributeName="Actions"
AttributeNamespace="http://www.oasis-
open.org/committees/security/docs/draft-sstc-schema-assertion-24">
<saml:AttributeValue>read</saml:AttributeValue>
</saml:Attribute>
</equal>
<equal>
<saml:AttributeDesignator AttributeName="requestor"
AttributeNamespace="//oasis-open.org/committees/xacml/docs/identifiers/" />
<saml:AttributeDesignator AttributeName="patientName"
AttributeNamespace="//medico.com/record/patient/" />
</equal>
</and>
</condition>
```

757 In the case where the *policy target* is computed as the intersection of the *targets* of the individual *rules*, the
758 *targets* may be omitted from the individual *rules*.

759 In the case that a *rule target* is present, the *rule* is evaluated according to the truth table of Table 1.

760 4.2.2.2. Rule-combining algorithm

761 The *rule-combining algorithm* specifies the algorithm by which the results of evaluating the component
762 *rules* are combined, when evaluating the *policy*.

763 The result of evaluating the *policy* is defined by the *rule-combining algorithm*. In the case that the *PDP* uses
764 a *policy* to determine its response to a *decision request*, the `saml:Decision` value is the value of the
765 *policy*, as defined by the *rule-combining algorithm*.

766 See Section 7 for an example of a *rule-combining algorithm*.

767 4.2.2.3. Obligations

768 The XACML `<rule>` syntax does not contain an element suitable for carrying *obligations*, therefore, if
769 required in a *policy*, *obligations* must be added by the writer of the *policy*.

770 When a *PDP* evaluates a *policy* containing *obligations*, it returns certain of those *obligations* to the *PEP* in its
771 *authorization decision*. The *obligations* that it returns to the *PEP* are those whose `xacml:fulfilOn`
772 attributes have the same value as the result of evaluating the *policy*.

773 4.2.2.4. Example policy statement

774 This section uses the example of Section 3 to illustrate the process of combining rules. The policy governing
775 read access to medical elements of a record is formed from each of the four rules. In plain language, the
776 combined rule is:

- 777 • Either the requestor is the patient; or
- 778 • the requestor is the parent or guardian and the patient is under 16; or
- 779 • the requestor is the primary care physician and a notification is sent to the patient; and
- 780 • the requestor is not an administrator.

781 The following XACML <policyStatement> illustrates the combined rules. Rules 1 and 4 are included by
782 reference, rule 2 is included as a digest, and rule 3 is explicitly included.

```
783 <?xml version="1.0" encoding="UTF-8"?>
784 <saml:Assertion MajorVersion="0" MinorVersion="28" AssertionID="A7F34K90"
785 Issuer="medico.com" IssueInstant="2002-03-22T08:23:47-05:00">
786   <policyStatement policyId="//medico.com/rules/policy5"
787     ruleCombiningAlgId="//www.oasis-
788     open.org/committees/xacml/docs/ruleCombiningAlgorithms/denyOverrides"
789     xmlns="http://www.oasis-open.org/committees/xacml/docs/draft-xacml-schema-
790     policy-12.xsd" xmlns:saml="http://www.oasis-
791     open.org/committees/security/docs/draft-sstc-schema-assertion-28.xsd"
792     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
793     xsi:schemaLocation="http://www.oasis-open.org/committees/xacml/docs/draft-
794     xacml-schema-policy-12.xsd
795     D:\MYDOCU~1\Standards\XACML\V12SCH~1\XACMLV~3.XSD">
796     <target>
797       <subjects>
798         <saml:Attribute AttributeName="role"
799         AttributeNamespace="//medico.com">
800           <saml:AttributeValue>*</saml:AttributeValue>
801         </saml:Attribute>
802       </subjects>
803       <resources>
804         <saml:Attribute AttributeName="documentURI"
805         AttributeNamespace="//medico.com">
806           <saml:AttributeValue>//medico.com/record/medical.*</saml:AttributeValue>
807         </saml:Attribute>
808       </resources>
809       <actions>
810         <saml:Action>read</saml:Action>
811       </actions>
812     </target>
813   </ruleSet>
814   <ruleSet>
815     <ruleDesignator>
816       <ruleId>//medico.com/rules/rule1</ruleId>
817     </ruleDesignator>
818     <ruleDesignator>
819       <ruleDigest
820       base64Digest="H7jiE0+jwkn63k/JhB3+D9aI4V3J9z/o0"/>
821     </ruleDesignator>
822     <rule ruleId="//medico.com/rules/rule3" effect="Permit">
823       <description>A physician may write any medical element for
824       which he or she is the designated primary care physician</description>
825       <target>
826         <subjects>
827           <saml:Attribute AttributeName="role"
828           AttributeNamespace="//medico.com">
829             <saml:AttributeValue>physician</saml:AttributeValue>
830           </saml:Attribute>
831         </subjects>
```

```

832         </subjects>
833         <resources>
834             <saml:Attribute AttributeName="documentURI"
835 AttributeNamespace="//medico.com">
836
837         <saml:AttributeValue>//medico.com/record/medical.*</saml:AttributeValue>
838         </saml:Attribute>
839         </resources>
840         <actions>
841             <saml:Action>write</saml:Action>
842         </actions>
843     </target>
844     <condition>
845         <and>
846             <equal>
847                 <saml:AttributeDesignator
848 AttributeName="requestor" AttributeNamespace="//oasis-
849 open.org/committees/xacml/docs/identifiers/" />
850                 <saml:AttributeDesignator
851 AttributeName="physicianName"
852 AttributeNamespace="//medico.com/record/primaryCarePhysician/" />
853             </equal>
854             <present>
855                 <saml:AttributeDesignator
856 AttributeName="patient/email" AttributeNamespace="//medico.com/record/" />
857             </present>
858         </and>
859     </condition>
860 </rule>
861 <ruleDesignator>
862     <ruleId>//medico.com/rules/rule4</ruleId>
863 </ruleDesignator>
864 </ruleSet>
865 <obligations>
866     <obligation ObligationId="//medico.com/emailer"
867 fulfilOn="Permit">
868         <saml:AttributeDesignator AttributeName="patient/email"
869 AttributeNamespace="//medico.com/record/" />
870         <saml:Attribute AttributeName="messageText"
871 AttributeNamespace="//medico.com/" />
872         <saml:AttributeValue>Your medical record has been accessed
873 by:</saml:AttributeValue>
874     </saml:Attribute>
875     <saml:AttributeDesignator AttributeName="requestor"
876 AttributeNamespace="//oasis-open.org/committees/xacml/docs/identifiers/" />
877 </obligation>
878 </obligations>
879 </policyStatement>
880 </saml:Assertion>

```

881 4.2.3. Policy set statement

882 A *policy set* comprises four main components:

- 883 • a *target*;
- 884 • a set of *policy statements*;
- 885 • *obligations*; and
- 886 • a *policy-combining algorithm*-identifier.

887 The *target* and *policy statement* components are to be interpreted as described above.

888 **4.2.3.1. Obligations**

889 The writer of a *policy set statement* MAY add *obligations* to the *policy set*, in addition to those contained in
890 the component *policies* and *policy sets*.

891 **4.2.3.2. Policy-combining algorithm**

892 The *policy-combining algorithm* is the algorithm by which the results of evaluating the component *policies*
893 are combined to form the value of the *policy set*. In the case that the *PDP* uses a *policy set* to determine its
894 response to a *decision request*, the `saml:Decision` value is the result of evaluating the *policy set*.

895 When a *PDP* evaluates a *policy set* containing *obligations*, it returns certain of those *obligations* to the *PEP*
896 in its *authorization decision*. The XACML `<obligation>` elements that are returned to the *PEP* are those
897 whose `xacml:fulfilOn` attributes have the same value as the result of evaluating the *policy set*.

898 As a consequence of this procedure, no *obligations* are returned to the *PEP* if the *policies* from which they
899 are drawn are not evaluated or their evaluated result is Indeterminate or Not applicable.

900 See Section 7 for an example of a *policy-combining algorithm*.

901 **5. Policy syntax (normative, with the exception of the schema**
902 **fragments)**

903 **5.1. Element `<policySetStatement>`**

904 The `<policySetStatement>` element is a top-level element in the XACML schema.

905 `<xs:element name="policySetStatement" type="xacml:PolicySetStatementType"/>`

906 **5.2. Element `<policyStatement>`**

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

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

909 **5.3. Element `<rule>`**

910 The `<rule>` element is a top-level element in the XACML schema.

911 `<xs:element name="rule" type="xacml:RuleType"/>`

912 **5.4. Element `<authorizationDecisionStatement>`**

913 The `<authorizationDecisionStatement>` element is a top-level element in the XACML schema.

914 `<xs:element name="authorizationDecisionStatement"`
915 `type="xacml:AuthorizationDecisionStatementType"/>`

916 **5.5. Complex type `PolicySetStatementType`**

917 Elements of type `PolicySetStatementType` extend the `saml:StatementAbstractType` so that they MAY be
918 included in a `<saml:Assertion>` element. The `<saml:Assertion>` element contains some policy meta-data.
919 The main elements of this type definition are the `<target>`, `<policySet>` and `<obligations>` elements and the
920 `policyCombiningAlgId` attribute. The `<policySet>` element SHALL contain references to the set of
921 policies that are to be combined in a policy set. The `<target>` element MAY be declared by the creator of
922 elements of this type, or it MAY be computed from the `<target>` elements of the referenced
923 `<policyStatement>` elements, either as an intersection or as a union. The `<obligations>` element SHALL

924 contain the set of <obligation> elements that MUST be discharged by the PEP. The
925 policyCombiningAlgId attribute SHALL contain a reference to the policy-combining algorithm by
926 which the referenced <policyStatement> elements MUST be combined.

```
927 <xs:complexType name="PolicySetStatementType">
928   <xs:complexContent>
929     <xs:extension base="saml:StatementAbstractType">
930       <xs:sequence>
931         <xs:element name="description" type="xs:string" minOccurs="0"/>
932         <xs:element name="target" type="xacml:TargetType"/>
933         <xs:element name="policySet" type="xacml:PolicySetType"
934 maxOccurs="unbounded"/>
935         <xs:element name="obligations" type="xacml:ObligationsType"
936 minOccurs="0"/>
937       </xs:sequence>
938       <xs:attribute name="policySetId" type="xs:anyURI" use="required"/>
939       <xs:attribute name="policySetName" type="xs:string" use="optional"/>
940       <xs:attribute name="policyCombiningAlgId" type="xs:anyURI" use="required"/>
941     </xs:extension>
942   </xs:complexContent>
943 </xs:complexType>
```

944 **5.6. Complex type PolicyStatementType**

945 Elements of type PolicyStatementType extend the saml:StatementAbstractType so that they MAY be
946 included in a <saml:Assertion> element. The <saml:Assertion> element contains some policy meta-data.
947 The main elements of this type definition are the <target>, <ruleSet> and <obligations> elements and the
948 ruleCombiningAlgId attribute. The <ruleSet> element SHALL contain references to the <rule>
949 elements that are to be combined in a policy. The <target> element MAY be declared by the creator of
950 elements of this type, or it MAY be computed from the <target> elements of the referenced <rule> elements,
951 either as an intersection or as a union. The <obligations> element SHALL contain the set of <obligation>
952 elements that MUST be discharged by the PEP. The ruleCombiningAlgId attribute SHALL contain a
953 reference to the rule-combining algorithm by which the <rule> elements MUST be combined.

```
954 <xs:complexType name="PolicyStatementType">
955   <xs:complexContent>
956     <xs:extension base="saml:StatementAbstractType">
957       <xs:sequence>
958         <xs:element name="description" type="xs:string" minOccurs="0"/>
959         <xs:element name="target" type="xacml:TargetType"/>
960         <xs:element name="ruleSet" type="xacml:RuleSetType"
961 maxOccurs="unbounded"/>
962         <xs:element name="obligations" type="xacml:ObligationsType"
963 minOccurs="0"/>
964       </xs:sequence>
965       <xs:attribute name="policyId" type="xs:anyURI" use="required"/>
966       <xs:attribute name="policyName" type="xs:string" use="optional"/>
967       <xs:attribute name="ruleCombiningAlgId" type="xs:anyURI" use="required"/>
968     </xs:extension>
969   </xs:complexContent>
970 </xs:complexType>
```

971 **5.7. Complex type RuleType**

972 The main elements of this type definition are the <target> and <condition> elements, and the effect
973 attribute.

```
974 <xs:complexType name="RuleType">
975   <xs:sequence>
976     <xs:element name="description" type="xs:string" minOccurs="0"/>
977     <xs:element name="target" type="xacml:TargetType" minOccurs="0"/>
978     <xs:element name="condition" type="xacml:PredicateExpressionType"
979 minOccurs="0"/>
980   </xs:sequence>
981   <xs:attribute name="ruleId" type="xs:anyURI" use="required"/>
982   <xs:attribute name="ruleName" type="xs:string" use="optional"/>
```

```
983     <xs:attribute name="effect" type="xacml:EffectType" use="required" />
984 </xs:complexType>
```

985 **5.8. Complex type AuthorizationDecisionStatementType**

986 The complex type AuthorizationDecisionStatementType extends the SAML
987 AuthorizationDecisionStatementType with the addition of an XACML <obligations> element.

```
988 <xs:complexType name="AuthorizationDecisionStatementType">
989   <xs:complexContent>
990     <xs:extension base="saml:AuthorizationDecisionStatementType">
991       <xs:sequence>
992         <xs:element name="obligations" type="xacml:ObligationsType" />
993       </xs:sequence>
994     </xs:extension>
995   </xs:complexContent>
996 </xs:complexType>
```

997 **5.9. Complex type TargetType**

998 Elements of this type identify the set of decision requests of type saml:AuthorizationDecisionQueryType
999 that the parent element is intended to evaluate. It contains definition for subjects, resources and actions. If
1000 the subject, resource and action identified in the decision request are contained within the target, then the
1001 parent element MAY be used to evaluate the request.

```
1002 <xs:complexType name="TargetType">
1003   <xs:sequence>
1004     <xs:element name="subjects" type="xacml:SubjectsType" />
1005     <xs:element name="resources" type="xacml:ResourcesType" />
1006     <xs:element name="actions" type="xacml:ActionTypes" />
1007   </xs:sequence>
1008 </xs:complexType>
```

1009 **5.10. Complex type SubjectType**

1010 Elements of type SubjectType identify a set of subjects by a list of <saml:Attribute> elements.

```
1011 <xs:complexType name="SubjectsType">
1012   <xs:sequence maxOccurs="unbounded">
1013     <xs:element ref="saml:Attribute" />
1014   </xs:sequence>
1015 </xs:complexType>
```

1016 **5.11. Complex type ResourceType**

1017 Elements of type ResourceType identify a set of resources by a list of <saml:Attribute> elements.

```
1018 <xs:complexType name="ResourcesType">
1019   <xs:sequence maxOccurs="unbounded">
1020     <xs:element ref="saml:Attribute" />
1021   </xs:sequence>
1022 </xs:complexType>
```

1023 **5.12. Complex type ActionType**

1024 Elements of type ActionType identify a set of actions by a set of <saml:Action> elements.

```
1025 <xs:complexType name="ActionTypes">
1026   <xs:sequence maxOccurs="unbounded">
1027     <xs:element ref="saml:Action" />
1028   </xs:sequence>
1029 </xs:complexType>
```

1030 **5.13. Simple type EffectType**

1031 Elements of this type derive from the saml:DecisionType. The only values supported in XACML are Permit
1032 and Deny.

```
1033 <xs:simpleType name="EffectType">  
1034 <xs:restriction base="saml:DecisionType">  
1035 <xs:enumeration value="Permit"/>  
1036 <xs:enumeration value="Deny"/>  
1037 </xs:restriction>  
1038 </xs:simpleType>
```

1039 **5.14. Complex type ObligationsType**

1040 Elements of type ObligationsType contain a set of XACML <obligation> elements.

```
1041 <xs:complexType name="ObligationsType">  
1042 <xs:sequence>  
1043 <xs:element name="obligation" type="xacml:ObligationType"  
1044 maxOccurs="unbounded"/>  
1045 </xs:sequence>  
1046 </xs:complexType>
```

1047 **5.15. Complex type ObligationType**

1048 Elements of type ObligationType contain an identifier for the obligation and a set of attributes that form
1049 arguments of the action defined by the obligation. The fulfilOn attribute indicates the authorization
1050 decision value for which this obligation must be fulfilled.

```
1051 <xs:complexType name="ObligationType">  
1052 <xs:choice maxOccurs="unbounded">  
1053 <xs:element ref="saml:AttributeDesignator"/>  
1054 <xs:element ref="saml:Attribute"/>  
1055 </xs:choice>  
1056 <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>  
1057 <xs:attribute name="fulfilOn" type="xacml:EffectType" use="required"/>  
1058 </xs:complexType>
```

1059 **5.16. Complex type PredicateExpressionType**

1060 Elements of type PredicateExpressionType contain either a <predicateExpression> or <predicate> element.
1061 These elements are both of abstract type. So, an element of this type MUST contain an element in the
1062 substitution group of <predicateExpression> or <predicate>.

```
1063 <xs:complexType name="PredicateExpressionType">  
1064 <xs:choice>  
1065 <xs:element ref="xacml:predicateExpression"/>  
1066 <xs:element ref="xacml:predicate"/>  
1067 </xs:choice>  
1068 </xs:complexType>
```

1069 **5.17. Element <predicateExpression>**

1070 The <predicateExpression> element is the head of a substitution group. Members of the group MUST derive
1071 from the type PredicateExpressionAbstractType. XACML defines a small number of predicate expressions
1072 such as <and> and <or>. Others MAY add their own expressions to this substitution group. In such a case,
1073 there is no guarantee of interoperability.

```
1074 <xs:element name="predicateExpression" type="xacml:PredicateExpressionAbstractType"  
1075 abstract="true"/>
```

1076 **5.18. Complex type PredicateExpressionAbstractType**

1077 PredicateExpressionAbstractType is an empty base type from which predicate expression types MAY be
1078 derived.

1079 `<xs:complexType name="PredicateExpressionAbstractType"/>`

1080 **5.19. Element <and>**

1081 The <and> element is an element of type AndType.

1082 `<xs:element name="and" type="xacml:AndType"`
1083 `substitutionGroup="xacml:predicateExpression"/>`

1084 **5.20. Element <or>**

1085 The <or> element is an element of type OrType.

1086 `<xs:element name="or" type="xacml:OrType"`
1087 `substitutionGroup="xacml:predicateExpression"/>`

1088 **5.21. Element <orderedOr>**

1089 The <orderedOr> element is an element of type OrderedOrType.

1090 `<xs:element name="orderedOr" type="xacml:OrderedOrType"`
1091 `substitutionGroup="xacml:predicateExpression"/>`

1092 **5.22. Element <nOf>**

1093 The <nOf> element is an element of type NOFType.

1094 `<xs:element name="nOf" type="xacml:NOFType"`
1095 `substitutionGroup="xacml:predicateExpression"/>`

1096 **5.23. Element <not>**

1097 The <not> element is an element of type NotType.

1098 `<xs:element name="not" type="xacml:NotType"`
1099 `substitutionGroup="xacml:predicateExpression"/>`

1100 **5.24. Complex type AndType**

1101 Elements of type AndType MAY be evaluated with range True, False and Indeterminate. They MAY contain
1102 any combination of <predicateExpression> and <predicate> elements. Elements of this type SHALL evaluate
1103 True if and only if all the elements contained in them evaluate True.

1104 `<xs:complexType name="AndType">`
1105 `<xs:choice minOccurs="0" maxOccurs="unbounded">`
1106 `<xs:element ref="xacml:predicateExpression"/>`
1107 `<xs:element ref="xacml:predicate"/>`
1108 `</xs:choice>`
1109 `</xs:complexType>`

1110 **5.25. Complex type OrType**

1111 Elements of type OrType MAY be evaluated with range True, False and Indeterminate. They MAY contain
1112 any combination of <predicateExpression> and <predicate> elements. Elements of this type SHALL evaluate
1113 True if one or more of the elements contained in them evaluates True.

1114 `<xs:complexType name="OrType">`
1115 `<xs:choice minOccurs="0" maxOccurs="unbounded">`
1116 `<xs:element ref="xacml:predicateExpression"/>`

```
1117     <xs:element ref="xacml:predicate"/>
1118   </xs:choice>
1119 </xs:complexType>
```

1120 **5.26. Complex type OrderedOrType**

1121 Elements of type OrderedOrType MAY be evaluated with range True, False and Indeterminate. They MAY
1122 contain any combination of <predicateExpression> and <predicate> elements. Elements of this type SHALL
1123 evaluate True if one or more of the elements contained in them evaluates True. The <predicateExpression>
1124 and <predicate> elements MUST be evaluated in the order in which they appear, and evaluation MAY halt
1125 once one has evaluated True.

```
1126 <xs:complexType name="OrderedOrType">
1127   <xs:choice minOccurs="0" maxOccurs="unbounded">
1128     <xs:element ref="xacml:predicateExpression"/>
1129     <xs:element ref="xacml:predicate"/>
1130   </xs:choice>
1131 </xs:complexType>
```

1132 **5.27. Complex type NofType**

1133 Elements of type NOfType MAY be evaluated with range True, False and Indeterminate. They MAY contain
1134 any combination of <predicateExpression> and <predicate> elements and a quorum attribute. Elements of
1135 this type SHALL evaluate True if and only if a number of the elements contained in them equal to the value
1136 of quorum evaluate True.

```
1137 <xs:complexType name="NofType">
1138   <xs:choice minOccurs="0" maxOccurs="unbounded">
1139     <xs:element ref="xacml:predicateExpression"/>
1140     <xs:element ref="xacml:predicate"/>
1141   </xs:choice>
1142   <xs:attribute name="quorum" type="xs:positiveInteger"/>
1143 </xs:complexType>
```

1144 **5.28. Complex type NotType**

1145 Elements of type NotType MAY be evaluated with range True, False and Indeterminate. They MAY contain
1146 a choice of a <predicateExpression> or a <predicate> element. Elements of this type SHALL evaluate True if
1147 and only if the element contained in them evaluates False.

```
1148 <xs:complexType name="NotType">
1149   <xs:choice>
1150     <xs:element ref="xacml:predicateExpression" minOccurs="0"/>
1151     <xs:element ref="xacml:predicate" minOccurs="0"/>
1152   </xs:choice>
1153 </xs:complexType>
```

1154 **5.29. Element <predicate>**

1155 The <predicate> element is the head of a substitution group. Members of the group MUST derive from the
1156 type PredicateAbstractType. XACML defines a small number of predicates such as <equal> and
1157 <greaterOrEqual>. Others MAY add their own expressions to this substitution group. In such a case, there is
1158 no guarantee of interoperability.

```
1159 <xs:element name="predicate" type="xacml:PredicateAbstractType" abstract="true"/>
```

1160 **5.30. Complex type PredicateAbstractType**

1161 PredicateAbstractType is an empty base type from which predicate expression types MAY be derived.

```
1162 <xs:complexType name="PredicateAbstractType"/>
```

1163 **5.31. Element <true>**

1164 The <true> element is in the substitution group of <predicate>. It is of type TrueType. It SHALL evaluate
 1165 True if and only if the attribute it references is true.

1166 `<xs:element name="true" type="xacml:TrueType" substitutionGroup="xacml:predicate"/>`

1167 **5.32. Element <present>**

1168 The <present> element is in the substitution group of <predicate>. It is of type PresentType. It SHALL
 1169 evaluate True if and only if the attribute it references is obtainable.

1170 `<xs:element name="present" type="xacml:PresentType"`
 1171 `substitutionGroup="xacml:predicate"/>`

1172 **5.33. Element <equal>**

1173 The <equal> element is in the substitution group of <predicate>. It is of type CompareType. It SHALL
 1174 evaluate to true if and only if the attributes it contains or references are equal.

1175 `<xs:element name="equal" type="xacml:CompareType" substitutionGroup="xacml:predicate"/>`

1176 Compatible type combinations are shown in Table 2.

First operand type	Second operand type
xs:integer, xs:positiveInteger, xs:negativeInteger, xs:nonNegativeInteger, xs:nonPositiveInteger, xs:int, xs:unsignedInt, xs:long, xs:unsignedLong, xs:short, xs:unsignedShort, xs:decimal, xs:float, xs:double	xs:integer, xs:positiveInteger, xs:negativeInteger, xs:nonNegativeInteger, xs:nonPositiveInteger, xs:int, xs:unsignedInt, xs:long, xs:unsignedLong, xs:short, xs:unsignedShort, xs:decimal, xs:float, xs:double
xs:time, xs:dateTime, xs:date	xs:time, xs:dateTime, xs:date
xs:duration	xs:duration
xs:gMonth	xs:gMonth
xs:gYear	xs:gYear
xs:gYearMonth	xs:gYearMonth
xs:gDay	xs:gDay
xs:gMonthDay	xs:gMonthDay

1177 **Table 2 - Compatible type combinations for the equal, greaterOrEqual and lessOrEqual**
 1178 **predicate attributes**

1179 **5.34. Element <greaterOrEqual>**

1180 The <greaterOrEqual> element is in the substitution group of <predicate>. It is of type CompareType. It
 1181 SHALL evaluate True if and only if the attribute value referenced by the first element of the CompareType is
 1182 greater than or equal to the attribute value referenced by the second element.

1183 `<xs:element name="greaterOrEqual" type="xacml:CompareType"`
 1184 `substitutionGroup="xacml:predicate"/>`

1185 Compatible type combinations are shown in Table 2.

1186 **5.35. Element <lessOrEqual>**

1187 The <lessOrEqual> element is in the substitution group of <predicate>. It is of type CompareType. It
1188 SHALL evaluate True if and only if the attribute value referenced by the first element of the CompareType is
1189 less than or equal to the attribute value referenced by the second element.

```
1190 <xs:element name="lessOrEqual" type="xacml:CompareType"  
1191 substitutionGroup="xacml:predicate"/>
```

1192 Compatible type combinations are shown in Table 2.

1193 **5.36. Element <subset>**

1194 The <subset> element is in the substitution group of <predicate>. It is of type CompareType. It SHALL
1195 evaluate True if and only if the set of attribute values referenced by the first element of the CompareType is
1196 amongst the set of attribute values referenced by the second element. Both sets of attributes SHALL be of
1197 type xsi:list. Individual attributes MAY be of any type. Equality of attribute values SHALL be determined
1198 by a simple string match.

```
1199 <xs:element name="subset" type="xacml:CompareType" substitutionGroup="xacml:predicate"/>
```

1200 **5.37. Element <superset>**

1201 The <superset> element is in the substitution group of <predicate>. It is of type CompareType. It SHALL
1202 evaluate True if and only if the attribute value referenced by the first element of the CompareType contains
1203 the set of attribute values referenced by the second element. Both sets of attributes SHALL be of type xsi:list.
1204 Individual attributes MAY be of any type. Equality of attribute values SHALL be determined by a simple
1205 string match.

```
1206 <xs:element name="supersetOf" type="xacml:CompareType"  
1207 substitutionGroup="xacml:predicate"/>
```

1208 **5.38. Element <patternMatch>**

1209 The <patternMatch> element is in the substitution group of <predicate>. It is of type CompareType. It
1210 SHALL evaluate True if and only if the attribute value referenced by the first element of the CompareType
1211 matches the pattern defined in the string referenced by the second element. The string of the second element
1212 SHALL be in the form of a regular expression. The first element SHALL be of one of the following types:
1213 xs:integer, xs:positiveInteger, xs:negativeInteger, xs:nonNegativeInteger, xs:nonPositiveInteger, xs:int,
1214 xs:unsignedInt, xs:long, xs:unsignedLong, xs:short, xs:unsignedShort, xs:decimal, xs:float, xs:double,
1215 xs:time, xs:dateTime, xs:date, xs:gMonth, xs:gYear, xs:gYearMonth, xs:gDay, xs:gMonthDay, xs:token,
1216 xs:NAME, xs:QNAME and xs:anyURI.

```
1217 <xs:element name="patternMatch" type="xacml:CompareType"  
1218 substitutionGroup="xacml:predicate"/>
```

1219 **5.39. Element <nonNullSetIntersection>**

1220 The <nonNullSetIntersection> element is in the substitution group of <predicate>. It contains an element of
1221 type CompareType. It SHALL evaluate True if and only if the set of attribute values referenced by the two
1222 elements of the CompareType have at least one value in common. Both sets of attributes SHALL be of type
1223 xsi:list. Individual attributes MAY be of any type. Equality of attribute values SHALL be determined by a
1224 simple string match.

```
1225 <xs:element name="nonNullSetIntersection" type="xacml:CompareType"  
1226 substitutionGroup="xacml:predicate"/>
```

1227 **5.40. Complex type TrueType**

1228 Elements of type TrueType contain an element of type <saml:AttributeDesignator>. It SHALL evaluate True
1229 if and only if the attribute obtained by resolving the <saml:AttributeDesignator> element is true.

```
1230 <xs:complexType name="TrueType">  
1231   <xs:complexContent>  
1232     <xs:extension base="xacml:PredicateAbstractType">  
1233       <xs:sequence>  
1234         <xs:element ref="saml:AttributeDesignator" />  
1235       </xs:sequence>  
1236     </xs:extension>  
1237   </xs:complexContent>  
1238 </xs:complexType>
```

1239 **5.41. Complex type PresentType**

1240 Elements of type PresentType contain an element of type <saml:AttributeDesignator>. It SHALL evaluate
1241 True if and only if the attribute obtained by resolving the <saml:AttributeDesignator> element exists.

```
1242 <xs:complexType name="PresentType">  
1243   <xs:complexContent>  
1244     <xs:extension base="xacml:PredicateAbstractType">  
1245       <xs:sequence>  
1246         <xs:element ref="saml:AttributeDesignator" />  
1247       </xs:sequence>  
1248     </xs:extension>  
1249   </xs:complexContent>  
1250 </xs:complexType>
```

1251 **5.42. Complex type CompareType**

1252 Elements of type CompareType contain two elements. Either may be an attribute type designator, an attribute
1253 type/value pair or an attribute function. At least one of the attributes SHOULD NOT be an attribute
1254 type/value pair or function containing only attribute type/value pairs. Otherwise, the result is entirely
1255 determined by the policy writer, independent of any attribute values provided by, or referenced in, the
1256 decision request.

```
1257 <xs:complexType name="CompareType">  
1258   <xs:complexContent>  
1259     <xs:extension base="xacml:PredicateAbstractType">  
1260       <xs:choice minOccurs="2" maxOccurs="2">  
1261         <xs:element ref="saml:AttributeDesignator" />  
1262         <xs:element ref="saml:Attribute" />  
1263         <xs:element ref="xacml:attributeFunction" />  
1264       </xs:choice>  
1265     </xs:extension>  
1266   </xs:complexContent>  
1267   <!-- XML operands in "set" operations MUST be of xsi:type xs:list -->  
1268   <!-- XML operands in "inequality" operations MUST contain an xsi:type attribute for  
1269   which  
1270   XACML defines a comparison algorithm -->  
1271 </xs:complexType>
```

1272 **5.43. Element <attributeFunction>**

1273 The <attributeFunction> element is the head of a substitution group. Members of the group MUST derive
1274 from the type AttributeFunctionAbstractType. XACML defines a small number of arithmetic functions on
1275 attributes. Others MAY add their own function to this substitution group. In such a case there is no
1276 guarantee of interoperability.

```
1277 <xs:element name="attributeFunction" type="xacml:AttributeFunctionAbstractType"  
1278 abstract="true" />
```

1279 **5.44. Complex type *AttributeFunctionAbstractType***

1280 *AttributeFunctionAbstractType* is an empty base type from which attribute function types MAY be derived.
 1281 Specifically, the *ArgumentsListType* derives by extension from this type.

1282 `<xs:complexType name="AttributeFunctionAbstractType" />`

1283 **5.45. Element *<plus>***

1284 The *<plus>* element evaluates to the sum of it arguments' values.

1285 `<xs:element name="plus" type="xacml:ArgumentListType"`
 1286 `substitutionGroup="xacml:attributeFunction" />`

1287 Compatible type combinations are shown in Table 3.

First operand	Subsequent operand	Result
xs:time	xs:duration	xs:time
xs:dateTime	xs:duration	xs:dateTime
xs:duration	xs:duration	xs:duration
xs:date	xs:duration	xs:date
xs:gMonth	xs:duration	xs:gMonth
xs:gYear	xs:duration	xs:gYear
xs:gYearMonth	xs:duration	xs:gYearMonth
xs:gDay	xs:duration	xs:gDay
xs:gMonthDay	xs:duration	xs:gMonthDay

1288 **Table 3 - Compatible type combinations for the plus attributeFunction attributes**

1289 ISSUE: This table has to be completed for other XML schema types.

1290 **5.46. Element *<minus>***

1291 The *<minus>* element evaluates to the difference between its first argument's value and the sum of its
 1292 remaining arguments' values.

1293 `<xs:element name="minus" type="xacml:ArgumentListType"`
 1294 `substitutionGroup="xacml:attributeFunction" />`

1295 Compatible type combinations are shown in Table 4.

First operand	Second operand	Result
xs:time	xs:time	xs:duration
xs:dateTime	xs:dateTime	xs:duration
xs:duration	xs:duration	xs:duration

xs:date	xs:date	xs:duration
xs:gMonth	xs:gMonth	xs:duration
xs:gYear	xs:gYear	xs:duration
xs:gYearMonth	xs:gYearMonth	xs:duration
xs:gDay	xs:gDay	xs:duration
xs:gMonthDay	xs:gMonthDay	xs:duration

Table 4 - Compatible type combinations for the plus attributeFunction attributes

1296

1297 ISSUE: This table has to be completed for other XML schema types.

1298 **5.47. Element <times>**

1299 The <times> element evaluates to the product of its arguments' values.

1300 `<xs:element name="times" type="xacml:ArgumentListType"`
1301 `substitutionGroup="xacml:attributeFunction" />`

1302 **5.48. Element <divide>**

1303 The <divide> element evaluates to the ratio between its first argument's value and the product of its
1304 remaining arguments' values.

1305 `<xs:element name="divide" type="xacml:ArgumentListType"`
1306 `substitutionGroup="xacml:attributeFunction" />`

1307 **5.49. Element <setOrder>**

1308 The <setOrder> element evaluates to the number of elements in its (list) parameter.

1309 `<xs:element name="setOrder" type="xacml:ArgumentListType"`
1310 `substitutionGroup="xacml:attributeFunction" />`

1311 **5.50. Element <union>**

1312 The <union> element evaluates to the union of its arguments' values.

1313 `<xs:element name="union" type="xacml:ArgumentListType"`
1314 `substitutionGroup="xacml:attributeFunction" />`

1315 **5.51. Element <intersection>**

1316 The <intersection> element evaluates to the intersection of its arguments' values.

1317 `<xs:element name="intersection" type="xacml:ArgumentListType"`
1318 `substitutionGroup="xacml:attributeFunction" />`

1319 **5.52. Complex type ArgumentListType**

1320 Elements of type ArgumentListType SHALL contain a list of attribute type designators or attribute type/value
1321 pairs that form the argument list for an attribute function. All attributes contained in, or referenced by, an
1322 element of this type MUST be of compatible type.

1323 `<xs:complexType name="ArgumentListType">`

```

1324     <xs:complexContent>
1325         <xs:extension base="xacml:AttributeFunctionAbstractType">
1326             <xs:choice maxOccurs="unbounded">
1327                 <xs:element ref="saml:AttributeDesignator"/>
1328                 <xs:element ref="saml:Attribute"/>
1329             </xs:choice>
1330         </xs:extension>
1331     </xs:complexContent>
1332 </xs:complexType>

```

1333 **5.53. Complex type PolicySetType**

1334 Elements of type PolicySetType SHALL contain a set of <policyStatement> or <policySetStatement>
1335 elements, or designators of <policyStatement> or <policySetStatement> elements.

```

1336 <xs:complexType name="PolicySetType">
1337     <xs:choice maxOccurs="unbounded">
1338         <xs:element ref="xacml:policySetStatement"/>
1339         <xs:element ref="xacml:policyStatement"/>
1340         <xs:element name="policySetDesignator" type="xacml:PolicySetDesignatorType"/>
1341         <xs:element name="policyDesignator" type="xacml:PolicyDesignatorType"/>
1342     </xs:choice>
1343 </xs:complexType>

```

1344 **5.54. Complex type PolicySetDesignatorType**

1345 Elements of type PolicySetDesignatorType SHALL designate policy sets by identifier or by inclusion of a
1346 <saml:Assertion> element containing a <policySetStatement> element.

```

1347 <xs:complexType name="PolicySetDesignatorType">
1348     <xs:sequence>
1349         <xs:element name="policySetId" type="xs:anyURI" minOccurs="0"/>
1350         <xs:element name="policySetAssertion" type="saml:AssertionType"/>
1351     </xs:sequence>
1352 </xs:complexType>

```

1353 **5.55. Complex type PolicyDesignatorType**

1354 Elements of type PolicyDesignatorType SHALL designate policies by identifier or by inclusion of a
1355 <saml:Assertion> element containing a <policyStatement> element.

```

1356 <xs:complexType name="PolicyDesignatorType">
1357     <xs:sequence>
1358         <xs:element name="policyId" type="xs:anyURI" minOccurs="0"/>
1359         <xs:element name="policyAssertion" type="saml:AssertionType"/>
1360     </xs:sequence>
1361 </xs:complexType>

```

1362 **5.56. Complex type RuleSetType**

1363 Elements of type RuleSetType SHALL contain a set of <rule> or <ruleDesignator> elements.

```

1364 <xs:complexType name="RuleSetType">
1365     <xs:choice maxOccurs="unbounded">
1366         <xs:element ref="xacml:rule"/>
1367         <xs:element name="ruleDesignator" type="xacml:RuleDesignatorType"/>
1368     </xs:choice>
1369 </xs:complexType>

```

1370 **5.57. Complex type RuleDesignatorType**

1371 Elements of type RuleDesignatorType SHALL designate a rule by identifier or by digest.

```

1372 <xs:complexType name="RuleDesignatorType">
1373     <xs:sequence>
1374         <xs:element name="ruleId" type="xs:anyURI" minOccurs="0"/>

```

```
1375     <xs:element name="ruleDigest" minOccurs="0">
1376         <xs:complexType>
1377             <xs:attribute name="digestAlgId" type="xs:string" default="SHA-1"/>
1378             <xs:attribute name="base64Digest" type="xs:string"/>
1379         </xs:complexType>
1380     </xs:element>
1381 </xs:sequence>
1382 </xs:complexType>
```

1383 6. XACML identifiers (normative)

1384 This section defines standard identifiers for commonly-used entities. All XACML-defined identifiers have
1385 the common base:

1386 [//www.oasis-open.org/committees/xacml/docs/](http://www.oasis-open.org/committees/xacml/docs/)

1387 6.1. Requestor

1388 6.2. Time of day

1389 6.3. Attributes

1390 XACML-defined attributes are represented by an element of type <saml:AttributeDesignatorType>. It has
1391 two attributes: AttributeNamespace and AttributeName. All XACML-defined attributes have the following
1392 value for AttributeNamespace:

1393 [//www.oasis-open.org/committees/xacml/docs/attributes/](http://www.oasis-open.org/committees/xacml/docs/attributes/)

1394 6.3.1. X.500 distinguished name

1395 X.500 distinguished name attributes have the following value of AttributeName:

1396 [X500DN](#)

1397 6.3.2. Unix file-system path

1398 UNIX file-system path attributes have the following value of AttributeName:

1399 [UFS](#)

1400 6.3.3. Uniform resource identifier

1401 Uniform resource identifier attributes have the following value of AttributeName:

1402 [URI](#)

1403 6.4. Authentication locality

1404 6.5. Deny-overrides rule-combining algorithm

1405 The deny-overrides rule-combining algorithm has the following value for ruleCombiningAlgId:

1406 [//www.oasis-open.org/committees/xacml/docs/ruleCombiningAlgorithms/denyOverrides](http://www.oasis-open.org/committees/xacml/docs/ruleCombiningAlgorithms/denyOverrides)

1407 6.6. Deny-overrides policy-combining algorithm

1408 The deny-overrides policy-combining algorithm has the following value for policyCombiningAlgId:

1409 [//www.oasis-open.org/committees/xacml/docs/policyCombiningAlgorithms/denyOverrides](http://www.oasis-open.org/committees/xacml/docs/policyCombiningAlgorithms/denyOverrides)

1410 7. Combining algorithms (normative)

1411 This section contains a description of the rule-combining and policy-combining algorithms specified by
1412 XACML.

1413 7.1. Deny-overrides

1414 The following is a specification for the "deny-overrides" rule-combining algorithm. The identifier for this
1415 algorithm is given in Section 6.5.

1416 *In the entire set of rules to be evaluated, if any of the rules evaluates to*
1417 *"deny", then the rule combination is defined to evaluate to "deny" (that is,*
1418 *"deny" takes precedence, regardless of how many rules evaluate to "permit",*
1419 *and causes the whole combination to return "deny"). Any rule that evaluates*
1420 *to "indeterminate" (that is, its return status cannot be determined for any*
1421 *reason) has the same effect as a "deny" in that it causes the combination to*
1422 *return "deny". Finally, if none of the rules are found to be applicable to the*
1423 *request, the rule combination returns "notApplicable".*

1424 What follows is a pseudocode representation of how the above specification MAY be implemented. This is
1425 provided for illustrative and explanatory purposes.

```
1426 effect policy(rule[]){  
1427     atLeastOnePermit = false;  
1428     for( i=0; i<=noOfRules; i++ ){  
1429         if(rule[i] == deny){  
1430             return(deny);  
1431         }  
1432         if(rule[i] == indeterminate){  
1433             return(deny);  
1434         }  
1435         if(rule[i] == permit){  
1436             atLeastOnePermit = true;  
1437         }  
1438     }  
1439     if atLeastOnePermit {  
1440         return(permit);  
1441     }  
1442     else{  
1443         return(notApplicable);  
1444     }  
1445 }
```

1446 The following is a specification for the "deny-overrides" policy-combining algorithm. The identifier for this
1447 algorithm is given in Section 6.6.

1448 *In the entire set of policies to be evaluated, if any of the policies evaluates*
1449 *to "deny", then the policy combination is defined to evaluate to "deny" (that*
1450 *is, "deny" takes precedence, regardless of how many policies evaluate to*
1451 *"permit", and causes the whole combination to return "deny"). Any policy*
1452 *that evaluates to "indeterminate" (that is, its return status cannot be*
1453 *determined for any reason) has the same effect as a "deny" in that it causes*
1454 *the combination to return "deny". Finally, if none of the policies are found to*
1455 *be applicable to the request, the policy combination returns "notApplicable".*

1456 What follows is a pseudocode representation of how the above specification MAY be implemented. This is
1457 provided for illustrative and explanatory purposes.

```

1458 effect policySet(policy[]){
1459     atLeastOnePermit = false;
1460     for( i=0; i<=noOfPolicies; i++ ){
1461         if(policy[i] == deny){
1462             return(deny);
1463         }
1464         if(policy[i] == indeterminate){
1465             return(deny);
1466         }
1467         if(policy[i] == permit){
1468             atLeastOnePermit = true;
1469         }
1470     }
1471     if atLeastOnePermit {
1472         return(permit);
1473     } else{
1474         return(notApplicable);
1475     }
1476 }

```

1477 Obligations of the individual policies SHALL be combined as described in Section 4.2.2.3.

1478 **8. Profiles (normative but not mandatory to implement)**

1479 **8.1. XACML**

1480 Describes subsets of XACML appropriate to general classes of problem

1481 **8.2. SAML**

1482 Describes the subset of SAML that is relevant to XACML

1483 We need to specify SAML status codes for situations specific to XACML, such as:

- 1484 • *PDP* has no policy for the requested target
- 1485 • *PDP* cannot retrieve the required attributes

1486 A compliant SAML based PDP MUST reply to an SAML Authorization Decision Request with a SAML
1487 Authorization Decision in accordance with operational semantics of the PDP stated in Section 9.1.

1488 **8.3. XML Digital Signature**

1489 Describes how XACML instances shall be integrity-protected in the case where XML DSig is used. *PAPs*
1490 MAY sign XACML <policyStatement> elements. When a *PAP* combines <policyStatement> elements, it
1491 MAY sign the resulting <policySetStatement> element.

1492 **8.4. LDAP**

1493 The <policyStatement> and <policySetStatement> elements MAY be published by means of an LDAP
1494 repository. In this case, conformant implementations SHALL behave as described in this section.

1495 *Target* conforms to a data model. XACML does not specify the data model, but it MUST be agreed between
1496 the *PAP* and the *PDP*. The data model MUST be semi-hierarchical. That is, it MUST have one or more
1497 disjoint trees for *resources* and one or more disjoint trees for *subjects*. *Actions* are leaf nodes of the *resource*
1498 node to which they apply. Each level in the tree is identified with an *attribute* name. A "path" is a list of

1499 *attribute*-name/value pairs linking a node to the root. The form of a *target* is a set of paths, one or more for
1500 each tree in the data model.

1501 A node MAY have more than one *target* associated with it.

1502 An authorization *decision request* also specifies a set of paths by (directly or indirectly) providing *resource*,
1503 *subject* and *action attribute* values.

1504 A *policy statement* is applicable to a *decision request* if and only if every path in its *target* is part of a path in
1505 the *decision request*.

1506 The DIT of the repository SHALL be congruent with that of the *target* data model. <policyStatement>
1507 elements shall be LDAP attributes of the entries at the lowest node of every path in the *target* data model. In
1508 practice, the *policy statements* may be referenced from these nodes rather than stored at them.

1509 When *policy statements* are combined in a *policy set statement*, the *policy set statement target* MUST be
1510 computed, and the repository must be updated. *Policy statements* that conform to different *target* data
1511 models MUST NOT be combined.

1512 The *policy set statement target* SHALL be computed by separately combining trees of the same type from
1513 each of the original *policy statement targets*. The combination may be in the form of a union or an
1514 intersection.

1515 A union combination retains all of the original paths. If, as the result, all possible paths containing a
1516 particular DIT node are retained, then the path may be truncated at that node.

1517 An intersection combination retains a path from one *target* if and only if it includes a path from the other
1518 *target*.

1519 The *policy set statement* SHOULD be stored at the lowest node of every retained path.

1520 Some SAML *attributes* have an internal tree structure (e.g. DNS names). A sub-tree of this structure SHALL
1521 be represented by a regular expression. When such an *attribute* defines a level in a *target* tree, the sub-tree
1522 defined by each node at that level SHALL be attached at that node.

1523 8.4.1. Overview

1524 XACML policies, or references to XACML policies, may be stored in entries for the resources, actions or
1525 subjects to which they relate. This directory schema defines an auxiliary object class (**xacmlPolicyInfo**) for
1526 adding XACML policy data to such entries as well as a directory attribute (**xacmlPolicyData**) to contain the
1527 policies or references in those entries.

1528 Alternatively, XACML policies may be stored in policy-specific entries and referenced from the resource,
1529 action and/or subject entries to which they relate. This schema defines a structural object class
1530 (**xacmlPolicyObject**) for defining such entries as well as a directory attribute (**xacmlPolicyRDN**) to contain the
1531 string used to name the policy entry in the directory. The **xacmlPolicyData** directory attribute is also used in
1532 these entries to contain the policies themselves.

1533 A PDP would use an LDAP Directory User Agent (DUA) to search the resources/subjects subtrees in the
1534 directory to find the resource, action or subject of interest and retrieve the **xacmlPolicyData** directory attribute
1535 from that entry. That attribute may contain the XACML policy or a pointer to another directory entry that
1536 contains the XACML policy. If it contains only a pointer, the PDP must query the directory again to retrieve
1537 the **xacmlPolicyData** directory attribute from the entry related to the pointer. The content of the pointer is the
1538 value of the **xacmlPolicyRDN** directory attribute that is the final Relative Distinguished Name (RDN) for the
1539 policy entry in the directory.

1540 8.4.2. Object Class Definitions

1541 The following object classes are defined for the LDAP profile for XACML.

1542 8.4.2.1 XACML Policy Info

1543 The **xacmlPolicyInfo** object class is used in defining entries for objects that hold XACML policy information
1544 in addition to other data, e.g. as part of a resource, action, or subject entry.

```
1545  
1546 xacmlPolicyInfo OBJECT-CLASS ::= {  
1547     SUBCLASS OF {top}  
1548     KIND auxiliary  
1549     MAY CONTAIN {xacmlPolicyData}  
1550     ID id-???-oc-xacmlPolicyInfo }
```

1551

1552 8.4.2.2 XACML Policy Object

1553 The **xacmlPolicyObject** object class is used in defining entries for objects that hold only XACML policy
1554 information.

```
1555  
1556 xacmlPolicyObject OBJECT-CLASS ::= {  
1557     SUBCLASS OF {top}  
1558     KIND structural  
1559     MUST CONTAIN {xacmlPolicyRDN}  
1560     MAY CONTAIN {xacmlPolicyData}  
1561     ID id-???-oc-xacmlPolicyObject }
```

1562

1563 The **xacmlPolicyRDN** directory attribute is used to name the entry and position it in a policy subtree.

1564

1565 8.4.3. Attribute Definitions

1566 The following directory attributes are defined for the LDAP profile for XACML.

1567 8.4.3.1 XACML Policy Data

1568 The **xacmlPolicyData** directory attribute is used to store XACML policy information.

```
1569  
1570 xacmlPolicyData ATTRIBUTE ::= {  
1571     WITH SYNTAX XacmlPolicySyntax  
1572     ID id-???-at-xacmlPolicyData }  
1573  
1574 XacmlPolicySyntax ::= SEQUENCE {  
1575     policyPointer [0] UTF8String OPTIONAL,  
1576     policyData [1] UTF8String OPTIONAL  
1577     -- at least one of the optional elements must be present-- }  
1578
```

1579 If **policyPointer** is present, it indicates the value of the **xacmlPolicyRDN** directory attribute that is used to form
1580 the final Relative Distinguished Name (RDN) of the entry that contains the actual policy information.

1581 If **policyData** is present, it contains the XACML <policyStatement> or <policySetStatement>.

1582

1583 8.4.3.2 XACML Policy RDN

1584 The **xacmlPolicyRDN** directory attribute is used to store the name of an **xacmlPolicyObject** entry relative to its
1585 position in the directory hierarchy.

```
1586  
1587        xacmlPolicyRDN ATTRIBUTE ::= {  
1588            WITH SYNTAX            UTF8String  
1589            EQUALITY MATCHING RULE xacmlPolicyRDNMatch  
1590            ID                    id-???-at-xacmlPolicyRDN }
```

1591

1592 8.4.4. Matching Rule Definitions

1593 The **xacmlPolicyRDNMatch** matching rule compares for equality a presented value with an attribute value of
1594 type **xacmlPolicyRDN**.

```
1595  
1596        xacmlPolicyRDNMatch    MATCHING-RULE ::= {  
1597            SYNTAX    UTF8String  
1598            ID        id-???-at-policyNameMatch }
```

1599
1600 This rule returns TRUE if the presented value is equal to the stored value of the **xacmlPolicyRDN** directory
1601 attribute.

1602

1603 **9. Operational Model (normative)**

1604 This section describes the operational model for an XACML-based environment.

1605 **9.1. Policy Decision Point (PDP)**

1606 Given a valid XACML "policy statement" or a "policy set statement", a compliant XACML PDP MUST
1607 evaluate that statement in accordance to the semantics specified in Sections 5, 6, and 7 when applied to an
1608 "authorization decision request". The PDP MUST return a "authorization decision", with one value of
1609 "permit", "deny", or "indeterminate". The PDP MAY return an "authorization decision" of "indeterminate"
1610 with an error code of "insufficient information", signifying that more information needed. In this case, the
1611 "authorization decision" MAY list the names of any attributes of the subject and the resource that are needed
1612 by the PDP to refine its "authorization decision".

1613 *Decision Convergence*

1614 A client of a PDP MAY resubmit a refined authorization decision request in response to an "authorization
1615 decision" of "indeterminate" with an error code of "insufficient information" by adding attribute values for
1616 the attribute names that are listed in the response.

1617 When the PDP returns an "authorization decision" of "indeterminate" with an error code of "insufficient
1618 information", a PDP MUST NOT list the names of any attribute of the subject or the resource of the
1619 "authorization decision request" for which values were already supplied in the "authorization decision
1620 request". Note, this requirement forces the PDP to eventually return an "authorization decision" of "permit",
1621 "deny", or "indeterminate" with some other reason, in response to successively refined "authorization
1622 decision requests".

1623

1624 **10. XACML extension points (non-normative)**

1625 Describes the points within the XACML model and schema where extensions can be added

1626 **10.1. Substitution groups**

1627 The following XACML elements are the heads of substitution groups.

- 1628 • Predicate,
1629 • predicateExpression, and
1630 • attributeFunction.

1631 Implementors MAY add elements that are derived from the appropriate type, wherever these elements appear
1632 in XACML.

1633 **10.2. URIs**

1634 The following XML attributes are URIs.

- 1635 • ruleCombiningAlgId,
1636 • policyCombiningAlgId,
1637 • saml:AttributeNameSpace and
1638 • saml:AttributeName.

1639 **11. Security and privacy (non-normative)**

1640 Vulnerabilities and safeguards

1641 **12. References**

1642 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
1643 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997

1644 **[RegEx]**

1645 **[LDAP]**

1646 **[SAML]** Security Assertion Markup Language available from [http://www.oasis-](http://www.oasis-open.org/committees/security/#documents)
1647 [open.org/committees/security/#documents](http://www.oasis-open.org/committees/security/#documents)

1648 **[XMLSig]** D. Eastlake et al., *XML-Signature Syntax and Processing*,
1649 <http://www.w3.org/TR/xmlsig-core/>, World Wide Web Consortium.

1650 **[XMLSig-XSD]** XML Signature Schema available from [http://www.w3.org/TR/2000/CR-](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd)
1651 [xmlsig-core-20001031/xmlsig-core-schema.xsd](http://www.w3.org/TR/2000/CR-xmlsig-core-20001031/xmlsig-core-schema.xsd).

13. Schema (normative)

This section contains the XACML schema definition.

```

1654 <?xml version="1.0" encoding="UTF-8"?>
1655 <!-- edited with XML Spy v4.1 U (http://www.xmlspy.com) by Tim Moses (private) -->
1656 <xs:schema targetNamespace="http://www.oasis-open.org/committees/xacml/docs/draft-xacml-
1657 schema-policy-13.xsd" xmlns:saml="http://www.oasis-open.org/committees/security/docs/draft-
1658 sstc-schema-assertion-28.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
1659 xmlns:xacml="http://www.oasis-open.org/committees/xacml/docs/draft-xacml-schema-policy-
1660 13.xsd" elementFormDefault="qualified" attributeFormDefault="unqualified">
1661   <xs:import namespace="http://www.oasis-open.org/committees/security/docs/draft-sstc-
1662 schema-assertion-28.xsd" schemaLocation="D:\My Documents\Standards\Xacml\v13 schema\draft-
1663 sstc-schema-assertion-28.xsd"/>
1664   <xs:element name="policySetStatement" type="xacml:PolicySetStatementType"/>
1665   <xs:element name="policyStatement" type="xacml:PolicyStatementType"/>
1666   <xs:element name="rule" type="xacml:RuleType"/>
1667   <xs:element name="authorizationDecisionStatement"
1668 type="xacml:AuthorizationDecisionStatementType"/>
1669   <xs:complexType name="PolicySetStatementType">
1670     <xs:complexContent>
1671       <xs:extension base="saml:StatementAbstractType">
1672         <xs:sequence>
1673           <xs:element name="description" type="xs:string" minOccurs="0"/>
1674           <xs:element name="target" type="xacml:TargetType"/>
1675           <xs:element name="policySet" type="xacml:PolicySetType"
1676 maxOccurs="unbounded"/>
1677           <xs:element name="obligations" type="xacml:ObligationsType"
1678 minOccurs="0"/>
1679         </xs:sequence>
1680         <xs:attribute name="policySetId" type="xs:anyURI" use="required"/>
1681         <xs:attribute name="policySetName" type="xs:string" use="optional"/>
1682         <xs:attribute name="policyCombiningAlgId" type="xs:anyURI" use="required"/>
1683       </xs:extension>
1684     </xs:complexContent>
1685   </xs:complexType>
1686   <xs:complexType name="PolicyStatementType">
1687     <xs:complexContent>
1688       <xs:extension base="saml:StatementAbstractType">
1689         <xs:sequence>
1690           <xs:element name="description" type="xs:string" minOccurs="0"/>
1691           <xs:element name="target" type="xacml:TargetType"/>
1692           <xs:element name="ruleSet" type="xacml:RuleSetType"
1693 maxOccurs="unbounded"/>
1694           <xs:element name="obligations" type="xacml:ObligationsType"
1695 minOccurs="0"/>
1696         </xs:sequence>
1697         <xs:attribute name="policyId" type="xs:anyURI" use="required"/>
1698         <xs:attribute name="policyName" type="xs:string" use="optional"/>
1699         <xs:attribute name="ruleCombiningAlgId" type="xs:anyURI" use="required"/>
1700       </xs:extension>
1701     </xs:complexContent>
1702   </xs:complexType>
1703   <xs:complexType name="RuleType">
1704     <xs:sequence>
1705       <xs:element name="description" type="xs:string" minOccurs="0"/>
1706       <xs:element name="target" type="xacml:TargetType" minOccurs="0"/>
1707       <xs:element name="condition" type="xacml:PredicateExpressionType"
1708 minOccurs="0"/>
1709     </xs:sequence>
1710     <xs:attribute name="ruleId" type="xs:anyURI" use="required"/>
1711     <xs:attribute name="ruleName" type="xs:string" use="optional"/>
1712     <xs:attribute name="effect" type="saml:DecisionType" use="required"/>
1713   </xs:complexType>
1714   <xs:complexType name="AuthorizationDecisionStatementType">
1715     <xs:complexContent>
1716       <xs:extension base="saml:AuthorizationDecisionStatementType">
1717         <xs:sequence>
1718           <xs:element name="obligations" type="xacml:ObligationsType"/>
1719         </xs:sequence>

```

```

1720         </xs:extension>
1721     </xs:complexContent>
1722 </xs:complexType>
1723 <xs:complexType name="TargetType">
1724     <xs:sequence>
1725         <xs:element name="subjects" type="xacml:SubjectsType"/>
1726         <xs:element name="resources" type="xacml:ResourcesType"/>
1727         <xs:element name="actions" type="xacml:ActionsType"/>
1728     </xs:sequence>
1729 </xs:complexType>
1730 <xs:complexType name="SubjectsType">
1731     <xs:sequence maxOccurs="unbounded">
1732         <xs:element ref="saml:Attribute"/>
1733     </xs:sequence>
1734 </xs:complexType>
1735 <xs:complexType name="ResourcesType">
1736     <xs:sequence maxOccurs="unbounded">
1737         <xs:element ref="saml:Attribute"/>
1738     </xs:sequence>
1739 </xs:complexType>
1740 <xs:complexType name="ActionsType">
1741     <xs:sequence maxOccurs="unbounded">
1742         <xs:element ref="saml:Action"/>
1743     </xs:sequence>
1744 </xs:complexType>
1745 <xs:complexType name="ObligationsType">
1746     <xs:sequence>
1747         <xs:element name="obligation" type="xacml:ObligationType"
1748 maxOccurs="unbounded"/>
1749     </xs:sequence>
1750 </xs:complexType>
1751 <xs:complexType name="ObligationType">
1752     <xs:choice maxOccurs="unbounded">
1753         <xs:element ref="saml:AttributeDesignator"/>
1754         <xs:element ref="saml:Attribute"/>
1755     </xs:choice>
1756     <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
1757     <xs:attribute name="fulfilOn" type="saml:DecisionType" use="required"/>
1758 </xs:complexType>
1759 <xs:complexType name="PredicateExpressionType">
1760     <xs:choice>
1761         <xs:element ref="xacml:predicateExpression"/>
1762         <xs:element ref="xacml:predicate"/>
1763     </xs:choice>
1764 </xs:complexType>
1765 <xs:element name="predicateExpression" type="xacml:PredicateExpressionAbstractType"
1766 abstract="true"/>
1767 <xs:complexType name="PredicateExpressionAbstractType"/>
1768 <xs:element name="and" type="xacml:AndType"
1769 substitutionGroup="xacml:predicateExpression"/>
1770 <xs:element name="or" type="xacml:OrType"
1771 substitutionGroup="xacml:predicateExpression"/>
1772 <xs:element name="orderedOr" type="xacml:OrderedOrType"
1773 substitutionGroup="xacml:predicateExpression"/>
1774 <xs:element name="nOf" type="xacml:NOfType"
1775 substitutionGroup="xacml:predicateExpression"/>
1776 <xs:element name="not" type="xacml:NotType"
1777 substitutionGroup="xacml:predicateExpression"/>
1778 <xs:complexType name="AndType">
1779     <xs:choice minOccurs="0" maxOccurs="unbounded">
1780         <xs:element ref="xacml:predicateExpression"/>
1781         <xs:element ref="xacml:predicate"/>
1782     </xs:choice>
1783 </xs:complexType>
1784 <xs:complexType name="OrType">
1785     <xs:choice minOccurs="0" maxOccurs="unbounded">
1786         <xs:element ref="xacml:predicateExpression"/>
1787         <xs:element ref="xacml:predicate"/>
1788     </xs:choice>
1789 </xs:complexType>
1790 <xs:complexType name="OrderedOrType">

```

```

1791     <xs:choice minOccurs="0" maxOccurs="unbounded">
1792         <xs:element ref="xacml:predicateExpression"/>
1793         <xs:element ref="xacml:predicate"/>
1794     </xs:choice>
1795 </xs:complexType>
1796 <xs:complexType name="NofType">
1797     <xs:choice minOccurs="0" maxOccurs="unbounded">
1798         <xs:element ref="xacml:predicateExpression"/>
1799         <xs:element ref="xacml:predicate"/>
1800     </xs:choice>
1801     <xs:attribute name="quorum" type="xs:positiveInteger"/>
1802 </xs:complexType>
1803 <xs:complexType name="NotType">
1804     <xs:choice>
1805         <xs:element ref="xacml:predicateExpression" minOccurs="0"/>
1806         <xs:element ref="xacml:predicate" minOccurs="0"/>
1807     </xs:choice>
1808 </xs:complexType>
1809 <xs:element name="predicate" type="xacml:PredicateAbstractType" abstract="true"/>
1810 <!--This is an XACML extensibility point.  New predicates may be added in the
1811 substitution group of "predicate"-->
1812 <xs:complexType name="PredicateAbstractType"/>
1813 <xs:element name="true" type="xacml:TrueType" substitutionGroup="xacml:predicate"/>
1814 <xs:element name="present" type="xacml:PresentType"
1815 substitutionGroup="xacml:predicate"/>
1816 <xs:element name="equal" type="xacml:CompareType" substitutionGroup="xacml:predicate"/>
1817 <xs:element name="greaterOrEqual" type="xacml:CompareType"
1818 substitutionGroup="xacml:predicate"/>
1819 <xs:element name="lessOrEqual" type="xacml:CompareType"
1820 substitutionGroup="xacml:predicate"/>
1821 <xs:element name="subset" type="xacml:CompareType" substitutionGroup="xacml:predicate"/>
1822 <xs:element name="superset" type="xacml:CompareType"
1823 substitutionGroup="xacml:predicate"/>
1824 <xs:element name="patternMatch" type="xacml:CompareType"
1825 substitutionGroup="xacml:predicate"/>
1826 <xs:element name="nonNullSetIntersection" type="xacml:CompareType"
1827 substitutionGroup="xacml:predicate"/>
1828 <xs:complexType name="TrueType">
1829     <xs:complexContent>
1830         <xs:extension base="xacml:PredicateAbstractType">
1831             <xs:sequence>
1832                 <xs:element ref="saml:AttributeDesignator"/>
1833             </xs:sequence>
1834         </xs:extension>
1835     </xs:complexContent>
1836 </xs:complexType>
1837 <xs:complexType name="PresentType">
1838     <xs:complexContent>
1839         <xs:extension base="xacml:PredicateAbstractType">
1840             <xs:sequence>
1841                 <xs:element ref="saml:AttributeDesignator"/>
1842             </xs:sequence>
1843         </xs:extension>
1844     </xs:complexContent>
1845 </xs:complexType>
1846 <xs:complexType name="CompareType">
1847     <xs:complexContent>
1848         <xs:extension base="xacml:PredicateAbstractType">
1849             <xs:choice minOccurs="2" maxOccurs="2">
1850                 <xs:element ref="saml:AttributeDesignator"/>
1851                 <xs:element ref="saml:Attribute"/>
1852                 <xs:element ref="xacml:attributeFunction"/>
1853             </xs:choice>
1854         </xs:extension>
1855     </xs:complexContent>
1856 </xs:complexType>
1857 <xs:element name="attributeFunction" type="xacml:AttributeFunctionAbstractType"
1858 abstract="true"/>
1859 <xs:complexType name="AttributeFunctionAbstractType"/>
1860 <xs:element name="plus" type="xacml:ArgumentListType"
1861 substitutionGroup="xacml:attributeFunction"/>

```

```

1862     <xs:element name="minus" type="xacml:ArgumentListType"
1863 substitutionGroup="xacml:attributeFunction" />
1864     <xs:element name="times" type="xacml:ArgumentListType"
1865 substitutionGroup="xacml:attributeFunction" />
1866     <xs:element name="divide" type="xacml:ArgumentListType"
1867 substitutionGroup="xacml:attributeFunction" />
1868     <xs:element name="setOrder" type="xacml:ArgumentListType"
1869 substitutionGroup="xacml:attributeFunction" />
1870     <xs:element name="union" type="xacml:ArgumentListType"
1871 substitutionGroup="xacml:attributeFunction" />
1872     <xs:element name="intersection" type="xacml:ArgumentListType"
1873 substitutionGroup="xacml:attributeFunction" />
1874     <xs:complexType name="ArgumentListType">
1875       <xs:complexContent>
1876         <xs:extension base="xacml:AttributeFunctionAbstractType">
1877           <xs:choice maxOccurs="unbounded">
1878             <xs:element ref="saml:AttributeDesignator" />
1879             <xs:element ref="saml:Attribute" />
1880             <xs:element ref="xacml:attributeFunction" />
1881           </xs:choice>
1882         </xs:extension>
1883       </xs:complexContent>
1884     </xs:complexType>
1885     <xs:complexType name="PolicySetType">
1886       <xs:choice maxOccurs="unbounded">
1887         <xs:element ref="xacml:policySetStatement" />
1888         <xs:element ref="xacml:policyStatement" />
1889         <xs:element name="policySetDesignator" type="xacml:PolicySetDesignatorType" />
1890         <xs:element name="policyDesignator" type="xacml:PolicyDesignatorType" />
1891       </xs:choice>
1892     </xs:complexType>
1893     <xs:complexType name="PolicySetDesignatorType">
1894       <xs:sequence>
1895         <xs:element name="policySetId" type="xs:anyURI" minOccurs="0" />
1896         <xs:element name="policySetAssertion" type="saml:AssertionType" />
1897       </xs:sequence>
1898     </xs:complexType>
1899     <xs:complexType name="PolicyDesignatorType">
1900       <xs:sequence>
1901         <xs:element name="policyId" type="xs:anyURI" minOccurs="0" />
1902         <xs:element name="policyAssertion" type="saml:AssertionType" />
1903       </xs:sequence>
1904     </xs:complexType>
1905     <xs:complexType name="RuleSetType">
1906       <xs:choice maxOccurs="unbounded">
1907         <xs:element ref="xacml:rule" />
1908         <xs:element name="ruleDesignator" type="xacml:RuleDesignatorType" />
1909       </xs:choice>
1910     </xs:complexType>
1911     <xs:complexType name="RuleDesignatorType">
1912       <xs:sequence>
1913         <xs:element name="ruleId" type="xs:anyURI" minOccurs="0" />
1914         <xs:element name="ruleDigest" minOccurs="0">
1915           <xs:complexType>
1916             <xs:attribute name="digestAlgId" type="xs:string" default="SHA-1" />
1917             <xs:attribute name="base64Digest" type="xs:string" />
1918           </xs:complexType>
1919         </xs:element>
1920       </xs:sequence>
1921     </xs:complexType>
1922 </xs:schema>

```

1923

1924

1925 14. Conformance (normative)

1926 Not the test cases themselves, but a description of how the test cases should be used. The test cases will be a
1927 set of files on the XACML Web site

1928 Conformance claims MAY be made by either one of two components in the XACML model:

1929 1. An implementation of a policy administration points that produces policy statements that conform with
1930 the XACML schema; and

1931 2. An implementation of a policy decision point that produces decisions in response to decision requests on
1932 the basis of XACML policy statements that conform with the XACML schema.

1933 In the current version of the specification, implementations of a policy retrieval point that produce policy
1934 statements that conform with the XACML schema by combining XACML applicable policies are treated in
1935 the same way as policy administration points, from the point of view of conformance.

1936 Policy administration points MAY claim conformance with the XACML specification provided merely that
1937 they produce schema-compliant policy statements.

1938 Policy decision points MAY claim conformance with the XACML specification provided that they correctly
1939 execute the XACML conformance test suite provided:

1940 <http://www.oasis-open.org/> ...

1941 XACML Test Suite

1942 The test suite comprises three directories:

1943 • Decision requests

1944 • Policies

1945 • Authentication and attribute assertions

1946 • Decision assertions

1947 The decision requests directory contains a set of text/xml/samlp files that are valid SAML authorization
1948 decision request messages.

1949 The policies directory contains precisely one XACML policy file whose target includes includes each of the
1950 decision requests.

1951 The assertions directory contains an unordered set of text/xml/saml files containing the attributes required to
1952 evaluate the policies in the policies directory.

1953 The decisions directory contains an unordered set of tect/xml/samlp files that are valid SAML authorization
1954 decision responses.

1955 A conformant XACML *PDP* implementation shall create a decision assertion in response to each and every
1956 decision request. The decision responses are linked to the corresponding decision requests by the request ID
1957 attribute.

1958 XACML implementations that target an application domain other than SAML may use a tool or process that
1959 is not an integral part of the implementation to convert between the SAML test vectors and its private data
1960 representation.

1961 Disclaimer: Implementors SHALL NOT consider the test cases provided in the XACML conformance test
1962 suite as providing 100% test coverage. OASIS does not represent that a conformant implementation will
1963 operate correctly in all respects nor that it is fit for its purpose.

1964

Appendix A. Notices

1965 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might
1966 be claimed to pertain to the implementation or use of the technology described in this document or the extent
1967 to which any license under such rights might or might not be available; neither does it represent that it has
1968 made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in
1969 OASIS specifications can be found at the OASIS website. Copies of claims of rights made available for
1970 publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a
1971 general license or permission for the use of such proprietary rights by implementors or users of this
1972 specification, can be obtained from the OASIS Executive Director.

1973 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
1974 other proprietary rights which may cover technology that may be required to implement this specification.
1975 Please address the information to the OASIS Executive Director.

1976 Copyright © The Organization for the Advancement of Structured Information Standards [OASIS] 2001. All
1977 Rights Reserved.

1978 This document and translations of it may be copied and furnished to others, and derivative works that
1979 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
1980 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
1981 this paragraph are included on all such copies and derivative works. However, this document itself may not
1982 be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed
1983 for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in
1984 the OASIS Intellectual Property Rights document must be followed, or as required to translate it into
1985 languages other than English.

1986 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or
1987 assigns.

1988 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1989 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1990 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY
1991 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A
1992 PARTICULAR PURPOSE.