

eXtensible Access Control Markup Language

(XACML) Version 2.0

1

4 OASIS Standard, 1 Feb 2005

5	Document Identifier: oasis-access_control-xacml-2.0-core-spec-os
6	Location: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf
7 8	Editor: Tim Moses, Entrust Inc. (tim.moses@entrust.com)
9	Abstract:
10	This specification defines version 2.0 of the extensible access-control markup language.
11	Status:
12 13	This version of the specification is an approved OASIS Standard within the OASIS Access Control TC.
14 15 16	Access Control TC members should send comments on this specification to the xacml@lists.oasis-open.org list. Others may use the following link and complete the comment form: http://oasis-open.org/committees/comments/form.php?wg_abbrev=xacml.
17 18 19 20 21	For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Access Control TC web page (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml). For any errata page for this specification, please refer to the Access Control TC web page (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml).
23	The non-normative errata page for this specification is located at
24	www.oasis-open.org/committees/access-control.
25	Copyright © OASIS Open 2004-2005 All Rights Reserved.

Table of contents

27	1. Intro	duction (non-normative)	8
28	1.1.	Glossary	8
29	1.1.1	Preferred terms	8
30	1.1.2	Related terms	10
31	1.2.	Notation	10
32	1.3.	Schema organization and namespaces	10
33	2. Back	ground (non-normative)	11
34	2.1.	Requirements	11
35	2.2.	Rule and policy combining	12
36	2.3.	Combining algorithms	12
37	2.4.	Multiple subjects	13
38	2.5.	Policies based on subject and resource attributes	13
39	2.6.	Multi-valued attributes	14
40	2.7.	Policies based on resource contents	14
41	2.8.	Operators	14
42	2.9.	Policy distribution	15
43	2.10.	Policy indexing	15
44	2.11.	Abstraction layer	16
45	2.12.	Actions performed in conjunction with enforcement	16
46	3. Mod	els (non-normative)	16
47	3.1.	Data-flow model	16
48	3.2.	XACML context	16
49	3.3.	Policy language model	16
50	3.3.1	Rule	16
51	3.3.2	Policy	16
52	3.3.3	Policy set	16
53	4. Exar	mples (non-normative)	16
54	4.1.	Example one	16
55	4.1.1	Example policy	16
56	4.1.2	2 Example request context	16
57	4.1.3	B Example response context	16
58	4.2.	Example two	16
59	4.2.1	Example medical record instance	16
60	4.2.2	2 Example request context	16
61	4.2.3	B Example plain-language rules	16

62	4.2.4	4 Example XACML rule instances	16
63	5. Polic	cy syntax (normative, with the exception of the schema fragments)	16
64	5.1.	Element <policyset></policyset>	16
65	5.2.	Element <description></description>	16
66	5.3.	Element <policysetdefaults></policysetdefaults>	16
67	5.4.	Element <xpathversion></xpathversion>	16
68	5.5.	Element <target></target>	16
69	5.6.	Element <subjects></subjects>	16
70	5.7.	Element <subject></subject>	16
71	5.8.	Element <subjectmatch></subjectmatch>	16
72	5.9.	Element <resources></resources>	16
73	5.10.	Element <resource></resource>	16
74	5.11.	Element <resourcematch></resourcematch>	16
75	5.12.	Element <actions></actions>	16
76	5.13.	Element <action></action>	16
77	5.14.	Element <actionmatch></actionmatch>	16
78	5.15.	Element <environments></environments>	16
79	5.16.	Element <environment></environment>	16
80	5.17.	Element <environmentmatch></environmentmatch>	16
81	5.18.	Element <policysetidreference></policysetidreference>	16
82	5.19.	Element <policyidreference></policyidreference>	16
83	5.20.	Simple type VersionType	16
84	5.21.	Simple type VersionMatchType	16
85	5.22.	Element <policy></policy>	16
86	5.23.	Element <policydefaults></policydefaults>	16
87	5.24.	Element <combinerparameters></combinerparameters>	16
88	5.25.	Element <combinerparameter></combinerparameter>	16
89	5.26.	Element <rulecombinerparameters></rulecombinerparameters>	16
90	5.27.	Element <policycombinerparameters></policycombinerparameters>	16
91	5.28.	Element <policysetcombinerparameters></policysetcombinerparameters>	16
92	5.29.	Element <rule></rule>	16
93	5.30.	Simple type EffectType	16
94	5.31.	Element <variabledefinition></variabledefinition>	16
95	5.32.	Element <variablereference></variablereference>	16
96	5.33.	Element <expression></expression>	16
97	5.34.	Element <condition></condition>	16

98	5.35.	Element <apply></apply>	16
99	5.36.	Element <function></function>	16
100	5.37.	Complex type AttributeDesignatorType	16
101	5.38.	Element <subjectattributedesignator></subjectattributedesignator>	16
102	5.39.	Element <resourceattributedesignator></resourceattributedesignator>	16
103	5.40.	Element <actionattributedesignator></actionattributedesignator>	16
104	5.41.	Element <environmentattributedesignator></environmentattributedesignator>	16
105	5.42.	Element <attributeselector></attributeselector>	16
106	5.43.	Element <attributevalue></attributevalue>	16
107	5.44.	Element <obligations></obligations>	16
108	5.45.	Element <obligation></obligation>	16
109	5.46.	Element <attributeassignment></attributeassignment>	16
110	6. Con	text syntax (normative with the exception of the schema fragments)	16
111	6.1.	Element <request></request>	16
112	6.2.	Element <subject></subject>	16
113	6.3.	Element <resource></resource>	16
114	6.4.	Element <resourcecontent></resourcecontent>	16
115	6.5.	Element <action></action>	16
116	6.6.	Element <environment></environment>	16
117	6.7.	Element <attribute></attribute>	16
118	6.8.	Element <attributevalue></attributevalue>	16
119	6.9.	Element <response></response>	16
120	6.10.	Element <result></result>	16
121	6.11.	Element < Decision >	16
122	6.12.	Element <status></status>	16
123	6.13.	Element <statuscode></statuscode>	16
124	6.14.	Element <statusmessage></statusmessage>	16
125	6.15.	Element <statusdetail></statusdetail>	16
126	6.16.	Element <missingattributedetail></missingattributedetail>	16
127	7. Fund	ctional requirements (normative)	16
128	7.1.	Policy enforcement point	16
129	7.1.	1. Base PEP	16
130	7.1.2	2. Deny-biased PEP	16
131	7.1.3	3. Permit-biased PEP	16
132	7.2.	Attribute evaluation	16
133	7.2.	1. Structured attributes	16

134	7.2.2	2. Attribute bags	16
135	7.2.3	3. Multivalued attributes	16
136	7.2.4	4. Attribute Matching	16
137	7.2.5	5. Attribute Retrieval	16
138	7.2.6	6. Environment Attributes	16
139	7.3.	Expression evaluation	16
140	7.4.	Arithmetic evaluation	16
141	7.5.	Match evaluation	16
142	7.6.	Target evaluation	16
143	7.7.	VariableReference Evaluation	16
144	7.8.	Condition evaluation	16
145	7.9.	Rule evaluation	16
146	7.10.	Policy evaluation	16
147	7.11.	Policy Set evaluation	16
148	7.12.	Hierarchical resources	16
149	7.13.	Authorization decision	16
150	7.14.	Obligations	16
151	7.15.	Exception handling	16
152	7.15	5.1. Unsupported functionality	16
153	7.15	5.2. Syntax and type errors	16
154	7.15	5.3. Missing attributes	16
155	8. XAC	CML extensibility points (non-normative)	16
156	8.1.	Extensible XML attribute types	16
157	8.2.	Structured attributes	16
158	9. Secu	urity and privacy considerations (non-normative)	16
159	9.1.	Threat model	16
160	9.1.1	1. Unauthorized disclosure	16
161	9.1.2	2. Message replay	16
162	9.1.3	3. Message insertion	16
163	9.1.4	4. Message deletion	16
164	9.1.5	5. Message modification	16
165	9.1.6	6. NotApplicable results	16
166	9.1.7	7. Negative rules	16
167	9.2.	Safeguards	16
168	9.2.1	1. Authentication	16
169	9.2.2	2. Policy administration	16

170	9.2.3.	Confidentiality	16
171	9.2.4.	Policy integrity	16
172	9.2.5.	Policy identifiers	16
173	9.2.6.	Trust model	16
174	9.2.7.	Privacy	16
175	10. Conformar	nce (normative)	16
176	10.1. Introd	luction	16
177	10.2. Confo	ormance tables	16
178	10.2.1.	Schema elements	16
179	10.2.2.	Identifier Prefixes	16
180	10.2.3.	Algorithms	16
181	10.2.4.	Status Codes	16
182	10.2.5.	Attributes	16
183	10.2.6.	Identifiers	16
184	10.2.7.	Data-types	16
185	10.2.8.	Functions	16
186	11. Reference	s	16
187	Appendix A. D	Pata-types and functions (normative)	16
188	A.1. Introduc	ction	16
189	A.2. Data-ty	pes	16
190	A.3. Functio	ons	16
191	A.3.1	Equality predicates	16
192	A.3.2	Arithmetic functions	16
193	A.3.3	String conversion functions	16
194	A.3.4	Numeric data-type conversion functions	16
195	A.3.5	Logical functions	16
196	A.3.6	Numeric comparison functions	16
197	A.3.7	Date and time arithmetic functions	16
198	A.3.8	Non-numeric comparison functions	16
199	A.3.9	String functions	16
200	A.3.10	Bag functions	16
201	A.3.11	Set functions	16
202	A.3.12	Higher-order bag functions	16
203	A.3.13	Regular-expression-based functions	16
204	A.3.14	Special match functions	16
205	A.3.15	XPath-based functions	16

206	Α.:	3.16 Extension functions and primitive types	16
207	Append	lix B. XACML identifiers (normative)	16
208	B.1. X	KACML namespaces	16
209	B.2. A	Access subject categories	16
210	В.3.	Data-types	16
211	B.4. S	Subject attributes	16
212	B.6. R	Resource attributes	16
213	B.7. A	Action attributes	16
214	B.8. E	Environment attributes	16
215	B.9. S	Status codes	16
216	B.10.	Combining algorithms	16
217	Append	lix C. Combining algorithms (normative)	16
218	C.1.	Deny-overrides	16
219	C.2.	Ordered-deny-overrides	16
220	C.3.	Permit-overrides	16
221	C.4.	Ordered-permit-overrides	16
222	C.5.	First-applicable	16
223	C.6.	Only-one-applicable	16
224	Append	lix D. Acknowledgments	16
225	Append	lix E. Notices	16

1. Introduction (non-normative)

229	1.1. Glossary
230	1.1.1 Preferred terms
231	Access - Performing an action
232	Access control - Controlling access in accordance with a policy
233	Action - An operation on a resource
234 235	Applicable policy - The set of policies and policy sets that governs access for a specific decision request
236 237	Attribute - Characteristic of a subject, resource, action or environment that may be referenced in a predicate or target (see also – named attribute)
238 239 240	Authorization decision - The result of evaluating applicable policy , returned by the PDP to the PEP . A function that evaluates to "Permit", "Deny", "Indeterminate" or "NotApplicable", and (optionally) a set of obligations
241	Bag - An unordered collection of values, in which there may be duplicate values
242 243	Condition - An expression of predicates. A function that evaluates to "True", "False" or "Indeterminate"
244	Conjunctive sequence - a sequence of predicates combined using the logical 'AND' operation
245	Context - The canonical representation of a decision request and an authorization decision
246 247 248	Context handler - The system entity that converts decision requests in the native request format to the XACML canonical form and converts authorization decisions in the XACML canonical form to the native response format
249	Decision – The result of evaluating a rule, policy or policy set
250	Decision request - The request by a PEP to a PDP to render an authorization decision
251	Disjunctive sequence - a sequence of predicates combined using the logical 'OR' operation
252	Effect - The intended consequence of a satisfied rule (either "Permit" or "Deny")
253 254	Environment - The set of attributes that are relevant to an authorization decision and are independent of a particular subject, resource or action

255 256 257	Named attribute – A specific instance of an attribute, determined by the attribute name and type the identity of the attribute holder (which may be of type: subject, resource, action or environment) and (optionally) the identity of the issuing authority
258 259	Obligation - An operation specified in a policy or policy set that should be performed by the PEP in conjunction with the enforcement of an authorization decision
260 261	Policy - A set of rules, an identifier for the rule-combining algorithm and (optionally) a set of obligations. May be a component of a policy set
262	Policy administration point (PAP) - The system entity that creates a policy or policy set
263 264	Policy-combining algorithm - The procedure for combining the decision and obligations from multiple policies
265 266 267 268	Policy decision point (PDP) - The system entity that evaluates applicable policy and renders an authorization decision. This term is defined in a joint effort by the IETF Policy Framework Working Group and the Distributed Management Task Force (DMTF)/Common Information Model (CIM) in [RFC3198]. This term corresponds to "Access Decision Function" (ADF) in [ISO10181-3].
269 270 271 272 273	Policy enforcement point (PEP) - The system entity that performs access control , by making decision requests and enforcing authorization decisions . This term is defined in a joint effort by the IETF Policy Framework Working Group and the Distributed Management Task Force (DMTF)/Common Information Model (CIM) in [RFC3198]. This term corresponds to "Access Enforcement Function" (AEF) in [ISO10181-3].
274	Policy information point (PIP) - The system entity that acts as a source of attribute values
275 276	Policy set - A set of policies , other policy sets , a policy-combining algorithm and (optionally) a set of obligations . May be a component of another policy set
277	Predicate - A statement about attributes whose truth can be evaluated
278	Resource - Data, service or system component
279	Rule - A target, an effect and a condition. A component of a policy
280	Rule-combining algorithm - The procedure for combining decisions from multiple rules
281	Subject - An actor whose attributes may be referenced by a predicate
282 283	Target - The set of decision requests , identified by definitions for resource , subject and action , that a rule , policy or policy set is intended to evaluate
284 285 286 287 288 289 290	Type Unification - The method by which two type expressions are "unified". The type expressions are matched along their structure. Where a type variable appears in one expression it is then "unified" to represent the corresponding structure element of the other expression, be it another variable or subexpression. All variable assignments must remain consistent in both structures. Unification fails if the two expressions cannot be aligned, either by having dissimilar structure, or by having instance conflicts, such as a variable needs to represent both "xs:string" and "xs:integer". For a full explanation of type unification , please see [Hancock].

1.1.2 Related terms 291 292 In the field of access control and authorization there are several closely related terms in common 293 use. For purposes of precision and clarity, certain of these terms are not used in this specification. 294 For instance, the term *attribute* is used in place of the terms: group and role. 295 In place of the terms: privilege, permission, authorization, entitlement and right, we use the term 296 297 The term object is also in common use, but we use the term *resource* in this specification. 298 Requestors and initiators are covered by the term subject. 1.2. Notation 299 300 This specification contains schema conforming to W3C XML Schema and normative text to describe the syntax and semantics of XML-encoded policy statements. 301 302 The key words "MUST". "MUST NOT". "REQUIRED". "SHALL". "SHALL NOT". "SHOULD". "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be 303 304 interpreted as described in IETF RFC 2119 [RFC2119] 305 "they MUST only be used where it is actually required for interoperation or to limit behavior which has potential for causing harm (e.g., limiting retransmissions)" 306 307 These keywords are thus capitalized when used to unambiguously specify requirements over 308 protocol and application features and behavior that affect the interoperability and security of 309 implementations. When these words are not capitalized, they are meant in their natural-language 310 sense. 311 Listings of XACML schema appear like this. 312 313 [a01] Example code listings appear like this. 314 Conventional XML namespace prefixes are used throughout the listings in this specification to 315 stand for their respective namespaces as follows, whether or not a namespace declaration is 316 present in the example: 317 The prefix xacml: stands for the XACML policy namespace. 318 The prefix xacml-context: stands for the XACML context namespace. The prefix ds: stands for the W3C XML Signature namespace [DS]. 319 320 The prefix xs: stands for the W3C XML Schema namespace [XS]. The prefix xf: stands for the XQuery 1.0 and XPath 2.0 Function and Operators 321 322 specification namespace [XF]. 323 This specification uses the following typographical conventions in text: <XACMLElement>, <ns:ForeignElement>, Attribute, Datatype, OtherCode. Terms in italic bold-face are 324 325 intended to have the meaning defined in the Glossary.

1.3. Schema organization and namespaces

The XACML policy syntax is defined in a schema associated with the following XML namespace:

326

329 The XACML context syntax is defined in a schema associated with the following XML namespace:

330 urn:oasis:names:tc:xacml:2.0:context

331

338

339

340

341

342

343

344

345

346 347

348

349

350

351

352

353

354

358

367

2. Background (non-normative)

The "economics of scale" have driven computing platform vendors to develop products with very generalized functionality, so that they can be used in the widest possible range of situations. "Out of the box", these products have the maximum possible privilege for accessing data and executing software, so that they can be used in as many application environments as possible, including those with the most permissive security policies. In the more common case of a relatively restrictive security policy, the platform's inherent privileges must be constrained, by configuration.

The security policy of a large enterprise has many elements and many points of enforcement. Elements of policy may be managed by the Information Systems department, by Human Resources, by the Legal department and by the Finance department. And the policy may be enforced by the extranet, mail, WAN and remote-access systems; platforms which inherently implement a permissive security policy. The current practice is to manage the configuration of each point of enforcement independently in order to implement the security policy as accurately as possible. Consequently, it is an expensive and unreliable proposition to modify the security policy. And, it is virtually impossible to obtain a consolidated view of the safeguards in effect throughout the enterprise to enforce the policy. At the same time, there is increasing pressure on corporate and government executives from consumers, shareholders and regulators to demonstrate "best practice" in the protection of the information assets of the enterprise and its customers.

- For these reasons, there is a pressing need for a common language for expressing security policy. If implemented throughout an enterprise, a common policy language allows the enterprise to manage the enforcement of all the elements of its security policy in all the components of its information systems. Managing security policy may include some or all of the following steps: writing, reviewing, testing, approving, issuing, combining, analyzing, modifying, withdrawing, retrieving and enforcing policy.
- 355 XML is a natural choice as the basis for the common security-policy language, due to the ease with 356 which its syntax and semantics can be extended to accommodate the unique requirements of this 357 application, and the widespread support that it enjoys from all the main platform and tool vendors.

2.1. Requirements

- 359 The basic requirements of a policy language for expressing information system security policy are:
- To provide a method for combining individual *rules* and *policies* into a single *policy set* that applies to a particular *decision request*.
- To provide a method for flexible definition of the procedure by which *rules* and *policies* are combined.
- To provide a method for dealing with multiple *subjects* acting in different capacities.
- To provide a method for basing an *authorization decision* on *attributes* of the *subject* and *resource*.
 - To provide a method for dealing with multi-valued attributes.

- To provide a method for basing an *authorization decision* on the contents of an information *resource*.
- To provide a set of logical and mathematical operators on *attributes* of the *subject*, *resource* and *environment*.
- To provide a method for handling a distributed set of *policy* components, while abstracting the method for locating, retrieving and authenticating the *policy* components.
- To provide a method for rapidly identifying the *policy* that applies to a given action, based upon the values of *attributes* of the *subjects, resource* and *action*.
- To provide an abstraction-layer that insulates the policy-writer from the details of the application environment.
- To provide a method for specifying a set of actions that must be performed in conjunction with policy enforcement.
- The motivation behind XACML is to express these well-established ideas in the field of accesscontrol policy using an extension language of XML. The XACML solutions for each of these requirements are discussed in the following sections.

2.2. Rule and policy combining

- 384 The complete *policy* applicable to a particular *decision request* may be composed of a number of
- individual *rules* or *policies*. For instance, in a personal privacy application, the owner of the
- 386 personal information may define certain aspects of disclosure *policy*, whereas the enterprise that is
- the custodian of the information may define certain other aspects. In order to render an
- 388 authorization decision, it must be possible to combine the two separate policies to form the
- 389 single *policy* applicable to the request.

383

403

- 390 XACML defines three top-level policy elements: <Rule>, <Policy> and <PolicySet>. The
- 391 <Rule> element contains a Boolean expression that can be evaluated in isolation, but that is not
- intended to be accessed in isolation by a **PDP**. So, it is not intended to form the basis of an
- 393 authorization decision by itself. It is intended to exist in isolation only within an XACML PAP,
- 394 where it may form the basic unit of management, and be re-used in multiple *policies*.
- 395 The <Policy> element contains a set of <Rule> elements and a specified procedure for
- combining the results of their evaluation. It is the basic unit of **policy** used by the **PDP**, and so it is
- intended to form the basis of an *authorization decision*.
- 398 The <PolicySet> element contains a set of <Policy> or other <PolicySet> elements and a
- 399 specified procedure for combining the results of their evaluation. It is the standard means for
- 400 combining separate *policies* into a single combined *policy*.
- 401 Hinton et al [Hinton94] discuss the question of the compatibility of separate *policies* applicable to
- 402 the same *decision request*.

2.3. Combining algorithms

- 404 XACML defines a number of combining algorithms that can be identified by a
- 405 RuleCombiningAlqId or PolicyCombiningAlqId attribute of the <Policy> or <PolicySet>
- 406 elements, respectively. The *rule-combining algorithm* defines a procedure for arriving at an
- 407 authorization decision given the individual results of evaluation of a set of rules. Similarly, the

- 408 *policy-combining algorithm* defines a procedure for arriving at an *authorization decision* given
- 409 the individual results of evaluation of a set of *policies*. Standard combining algorithms are defined
- 410 for

439

- Deny-overrides (Ordered and Unordered),
- Permit-overrides (Ordered and Unordered),
- 413 First-applicable and
- 414 Only-one-applicable.
- 415 In the case of the Deny-overrides algorithm, if a single <Rule> or <Policy> element is
- 416 encountered that evaluates to "Deny", then, regardless of the evaluation result of the other <Rule>
- 417 or <Policy> elements in the *applicable policy*, the combined result is "Deny".
- 418 Likewise, in the case of the Permit-overrides algorithm, if a single "Permit" result is encountered,
- 419 then the combined result is "Permit".
- In the case of the "First-applicable" combining algorithm, the combined result is the same as the
- result of evaluating the first <Rule>, <Policy> or <PolicySet> element in the list of rules
- 422 whose *target* is applicable to the *decision request*.
- The "Only-one-applicable" *policy-combining algorithm* only applies to *policies*. The result of this
- combining algorithm ensures that one and only one *policy* or *policy set* is applicable by virtue of
- 425 their *targets*. If no *policy* or *policy set* applies, then the result is "NotApplicable", but if more than
- one *policy* or *policy* set is applicable, then the result is "Indeterminate". When exactly one *policy*
- 427 or *policy set* is applicable, the result of the combining algorithm is the result of evaluating the
- 428 single *applicable policy* or *policy set*.
- 429 **Policies** and **policy sets** may take parameters that modify the behaviour of the **combining**
- 430 *algorithms*. However, none of the standard *combining algorithms* is affected by parameters.
- Users of this specification may, if necessary, define their own combining algorithms.

2.4. Multiple subjects

- 433 Access-control policies often place requirements on the actions of more than one *subject*. For
- instance, the policy governing the execution of a high-value financial transaction may require the
- approval of more than one individual, acting in different capacities. Therefore, XACML recognizes
- 436 that there may be more than one **subject** relevant to a **decision request**. An **attribute** called
- 437 "subject-category" is used to differentiate between *subjects* acting in different capacities. Some
- 438 standard values for this *attribute* are specified, and users may define additional ones.

2.5. Policies based on subject and resource attributes

- Another common requirement is to base an *authorization decision* on some characteristic of the
- 441 **subject** other than its identity. Perhaps, the most common application of this idea is the **subject's**
- role [RBAC]. XACML provides facilities to support this approach. Attributes of subjects
- 443 contained in the request context may be identified by the <SubjectAttributeDesignator>
- 444 element. This element contains a URN that identifies the attribute. Alternatively, the
- 445
 445 <a href="
- identify a particular *subject attribute* value by its location in the *context* (see Section 2.11 for an
- 447 explanation of *context*).

- 448 XACML provides a standard way to reference the *attributes* defined in the LDAP series of
- specifications [LDAP-1, LDAP-2]. This is intended to encourage implementers to use standard
- 450 *attribute* identifiers for some common *subject attributes*.
- 451 Another common requirement is to base an *authorization decision* on some characteristic of the
- 452 **resource** other than its identity. XACML provides facilities to support this approach. **Attributes** of
- 453 the **resource** may be identified by the <ResourceAttributeDesignator> element. This
- 454 element contains a URN that identifies the *attribute*. Alternatively, the <attributeSelector>
- element may contain an XPath expression over the request *context* to identify a particular
- 456 **resource attribute** value by its location in the **context**.

2.6. Multi-valued attributes

- The most common techniques for communicating *attributes* (LDAP, XPath, SAML, etc.) support
- 459 multiple values per *attribute*. Therefore, when an XACML *PDP* retrieves the value of a *named*
- 460 **attribute**, the result may contain multiple values. A collection of such values is called a **bag**. A
- 461 **bag** differs from a set in that it may contain duplicate values, whereas a set may not. Sometimes
- this situation represents an error. Sometimes the XACML *rule* is satisfied if any one of the
- 463 **attribute** values meets the criteria expressed in the **rule**.
- 464 XACML provides a set of functions that allow a policy writer to be absolutely clear about how the
- 465 **PDP** should handle the case of multiple **attribute** values. These are the "higher-order" functions
- 466 (see Section A.3).

2.7. Policies based on resource contents

- In many applications, it is required to base an *authorization decision* on data *contained in* the
- 469 information *resource* to which *access* is requested. For instance, a common component of privacy
- 470 **policy** is that a person should be allowed to read records for which he or she is the subject. The
- 471 corresponding *policy* must contain a reference to the *subject* identified in the information *resource*
- 472 itself

467

478

457

- 473 XACML provides facilities for doing this when the information *resource* can be represented as an
- 474 XML document. The <a tributeSelector> element may contain an XPath expression over the
- 475 request *context* to identify data in the information *resource* to be used in the *policy* evaluation.
- 476 In cases where the information **resource** is not an XML document, specified **attributes** of the
- *resource* can be referenced, as described in Section 2.4.

2.8. Operators

- 479 Information security *policies* operate upon *attributes* of *subjects*, the *resource*, the *action* and
- 480 the **environment** in order to arrive at an **authorization decision**. In the process of arriving at the
- 481 authorization decision, attributes of many different types may have to be compared or computed.
- For instance, in a financial application, a person's available credit may have to be calculated by
- 483 adding their credit limit to their account balance. The result may then have to be compared with the
- 484 transaction value. This sort of situation gives rise to the need for arithmetic operations on
- 485 **attributes** of the **subject** (account balance and credit limit) and the **resource** (transaction value).
- 486 Even more commonly, a *policy* may identify the set of roles that are permitted to perform a
- particular action. The corresponding operation involves checking whether there is a non-empty
- intersection between the set of roles occupied by the *subject* and the set of roles identified in the
- 489 *policy*. Hence the need for set operations.

- 490 XACML includes a number of built-in functions and a method of adding non-standard functions.
- These functions may be nested to build arbitrarily complex expressions. This is achieved with the
- 492 <Apply> element. The <Apply> element has an XML attribute called FunctionId that identifies
- 493 the function to be applied to the contents of the element. Each standard function is defined for
- 494 specific argument data-type combinations, and its return data-type is also specified. Therefore,
- data-type consistency of the *policy* can be checked at the time the *policy* is written or parsed.
- 496 And, the types of the data values presented in the request *context* can be checked against the
- values expected by the *policy* to ensure a predictable outcome.
- In addition to operators on numerical and set arguments, operators are defined for date, time and
- 499 duration arguments.
- Relationship operators (equality and comparison) are also defined for a number of data-types,
- including the RFC822 and X.500 name-forms, strings, URIs, etc..
- Also noteworthy are the operators over Boolean data-types, which permit the logical combination of
- 503 predicates in a rule. For example, a rule may contain the statement that access may be
- 504 permitted during business hours AND from a terminal on business premises.
- The XACML method of representing functions borrows from MathML [MathML] and from the
- XQuery 1.0 and XPath 2.0 Functions and Operators specification [XF].

2.9. Policy distribution

- 508 In a distributed system, individual *policy* statements may be written by several policy writers and
- enforced at several enforcement points. In addition to facilitating the collection and combination of
- 510 independent *policy* components, this approach allows *policies* to be updated as required. XACML
- 511 policy statements may be distributed in any one of a number of ways. But, XACML does not
- describe any normative way to do this. Regardless of the means of distribution, *PDPs* are
- 513 expected to confirm, by examining the *policy's* <Target> element that the policy is applicable to
- 514 the **decision request** that it is processing.
- 515 <Policy> elements may be attached to the information *resources* to which they apply, as
- described by Perritt [Perritt93]. Alternatively, <Policy> elements may be maintained in one or
- 517 more locations from which they are retrieved for evaluation. In such cases, the *applicable policy*
- 518 may be referenced by an identifier or locator closely associated with the information *resource*.

2.10. Policy indexing

- 520 For efficiency of evaluation and ease of management, the overall security policy in force across an
- 521 enterprise may be expressed as multiple independent *policy* components. In this case, it is
- 522 necessary to identify and retrieve the *applicable policy* statement and verify that it is the correct
- one for the requested action before evaluating it. This is the purpose of the <Target> element in
- 524 XACML.

507

- 525 Two approaches are supported:
- 1. *Policy* statements may be stored in a database,. In this case, the *PDP* should form a database
- query to retrieve just those *policies* that are applicable to the set of *decision requests* to which it expects to respond. Additionally, the *PDP* should evaluate the <Target> element of
- 529 the retrieved *policy* or *policy set* statements as defined by the XACML specification.
- Alternatively, the *PDP* may be loaded with all available policies and evaluate their <Target> elements in the context of a particular *decision request*, in order to identify the *policies* and *policy sets* that are applicable to that request.

The use of constraints limiting the applicability of a *policy* were described by Sloman [Sloman94].

2.11. Abstraction layer

- 535 **PEPs** come in many forms. For instance, a **PEP** may be part of a remote-access gateway, part of a Web server or part of an email user-agent, etc.. It is unrealistic to expect that all **PEPs** in an
- enterprise do currently, or will in the future, issue *decision requests* to a *PDP* in a common format.
- Nevertheless, a particular *policy* may have to be enforced by multiple *PEPs*. It would be inefficient
- to force a policy writer to write the same *policy* several different ways in order to accommodate the
- format requirements of each *PEP*. Similarly attributes may be contained in various envelope types
- 541 (e.g. X.509 attribute certificates, SAML attribute assertions, etc.). Therefore, there is a need for a
- canonical form of the request and response handled by an XACML *PDP*. This canonical form is
- 543 called the XACML *context*. Its syntax is defined in XML schema.
- Naturally, XACML-conformant **PEPs** may issue requests and receive responses in the form of an
- 545 XACML context. But, where this situation does not exist, an intermediate step is required to
- 546 convert between the request/response format understood by the *PEP* and the XACML *context*
- format understood by the *PDP*.
- 548 The benefit of this approach is that *policies* may be written and analyzed independent of the
- specific environment in which they are to be enforced.
- In the case where the native request/response format is specified in XML Schema (e.g. a SAML-
- conformant *PEP*), the transformation between the native format and the XACML *context* may be
- specified in the form of an Extensible Stylesheet Language Transformation [XSLT].
- 553 Similarly, in the case where the **resource** to which **access** is requested is an XML document, the
- resource itself may be included in, or referenced by, the request context. Then, through the use
- of XPath expressions [XPath] in the *policy*, values in the *resource* may be included in the *policy*
- 556 evaluation.

557

567

569

533

534

2.12. Actions performed in conjunction with enforcement

- In many applications, policies specify actions that MUST be performed, either instead of, or in
- addition to, actions that MAY be performed. This idea was described by Sloman [Sloman94].
- 560 XACML provides facilities to specify actions that MUST be performed in conjunction with policy
- evaluation in the <Obligations> element. This idea was described as a provisional action by
- Kudo [Kudo00]. There are no standard definitions for these actions in version 2.0 of XACML.
- Therefore, bilateral agreement between a *PAP* and the *PEP* that will enforce its *policies* is required
- for correct interpretation. *PEPs* that conform with v2.0 of XACML are required to deny *access*
- unless they understand and can discharge all of the <Obligations> elements associated with the
- *applicable policy*. <Obligations> elements are returned to the *PEP* for enforcement.

3. Models (non-normative)

The data-flow model and language model of XACML are described in the following sub-sections.

3.1. Data-flow model

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

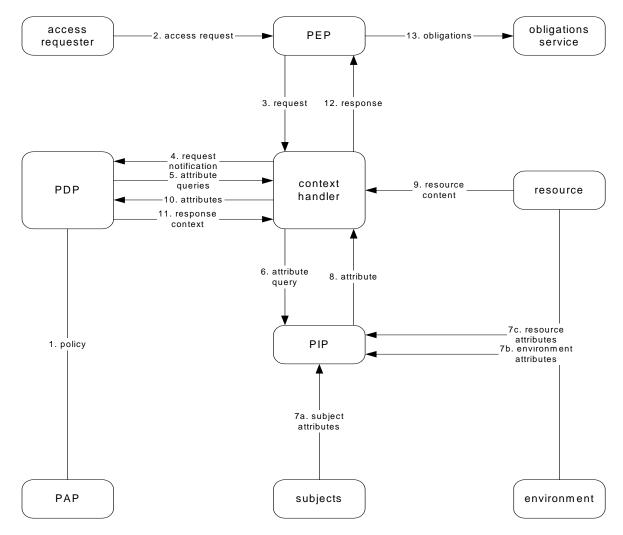


Figure 1 - Data-flow diagram

Note: some of the data-flows shown in the diagram may be facilitated by a repository. For instance, the communications between the *context* handler and the *PIP* or the communications between the *PDP* and the *PAP* 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.

578 The model operates by the following steps.

571

572

573

574

575 576

- 579 1. *PAP*s write *policies* and *policy sets* and make them available to the *PDP*. These *policies* or *policy sets* represent the complete policy for a specified *target*.
- 581 2. The access requester sends a request for access to the *PEP*.
- 582 3. The *PEP* sends the request for *access* to the *context handler* in its native request format, optionally including *attributes* of the *subjects*, *resource*, *action* and *environment*.
- 584 4. The *context handler* constructs an XACML request *context* and sends it to the *PDP*.
- 585 5. The *PDP* requests any additional *subject*, *resource*, *action* and *environment attributes* from the *context handler*.

- 587 6. The context handler requests the attributes from a **PIP**.
- 588 7. The **PIP** obtains the requested **attributes**.
- 589 8. The *PIP* returns the requested *attributes* to the *context handler*.
- 590 9. Optionally, the *context handler* includes the *resource* in the *context*.
- 591 10. The *context handler* sends the requested *attributes* and (optionally) the *resource* to the *PDP*.
 592 The *PDP* evaluates the *policy*.
- 593 11. The *PDP* returns the response *context* (including the *authorization decision*) to the *context* handler.
- 595 12. The *context handler* translates the response *context* to the native response format of the *PEP*. The *context handler* returns the response to the *PEP*.
- 597 13. The *PEP* fulfills the *obligations*.

601

602

603

604

605

606

607 608

609

610

611 612

613 614

617

14. (Not shown) If access is permitted, then the PEP permits access to the resource; otherwise, it denies access.

3.2. XACML context

XACML is intended to be suitable for a variety of application environments. The core language is insulated from the application environment by the XACML *context*, as shown in Figure 2, in which the scope of the XACML specification is indicated by the shaded area. The XACML *context* is defined in XML schema, describing a canonical representation for the inputs and outputs of the *PDP*. *Attributes* referenced by an instance of XACML policy may be in the form of XPath expressions over the *context*, or attribute designators that identify the *attribute* by *subject*, *resource*, *action* or *environment* and its identifier, data-type and (optionally) its issuer. Implementations must convert between the *attribute* representations in the application environment (e.g., SAML, J2SE, CORBA, and so on) and the *attribute* representations in the XACML *context*. How this is achieved is outside the scope of the XACML specification. In some cases, such as SAML, this conversion may be accomplished in an automated way through the use of an XSLT transformation.

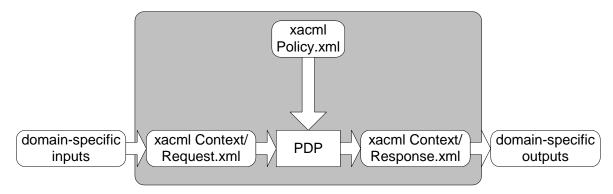


Figure 2 - XACML context

Note: The *PDP* is not required to operate directly on the XACML representation of a policy. It may operate directly on an alternative representation.

See Section 7.2.5 for a more detailed discussion of the request *context*.

3.3. Policy language model

- The policy language model is shown in Figure 3. The main components of the model are:
- 620 Rule;

618

- 621 *Policy*; and
- 622 Policy set.
- These are described in the following sub-sections.

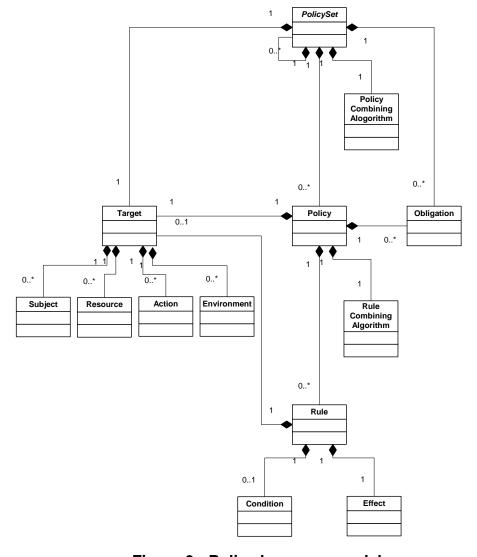


Figure 3 - Policy language model

3.3.1 Rule

624

625

627

628

A *rule* is the most elementary unit of *policy*. It may exist in isolation only *within* one of the major actors of the XACML domain. In order to exchange *rules* between major actors, they must be

- 629 encapsulated in a *policy*. A *rule* can be evaluated on the basis of its contents. The main 630 components of a *rule* are: 631 a target; 632 an effect and 633 a condition. 634 These are discussed in the following sub-sections. 635 **3.3.1.1.** Rule target 636 The target defines the set of: 637 resources: 638 subjects: 639 actions and 640 environment to which the rule is intended to apply. The <Condition> element may further refine the 641 642 applicability established by the *target*. If the *rule* is intended to apply to all entities of a particular 643 data-type, then the corresponding entity is omitted from the target. An XACML PDP verifies that 644 the matches defined by the target are satisfied by the subjects, resource, action and 645 environment attributes in the request context. Target definitions are discrete, in order that 646 applicable *rules* may be efficiently identified by the *PDP*. 647 The <Target> element may be absent from a <Rule>. In this case, the target of the <Rule> is 648 the same as that of the parent <Policy> element. 649 Certain subject name-forms, resource name-forms and certain types of resource are internally structured. For instance, the X.500 directory name-form and RFC 822 name-form are structured 650 651 subject name-forms, whereas an account number commonly has no discernible structure. UNIX file-system path-names and URIs are examples of structured resource name-forms. And an XML 652 653 document is an example of a structured resource.
- 654 Generally, the name of a node (other than a leaf node) in a structured name-form is also a legal
- instance of the name-form. So, for instance, the RFC822 name "med.example.com" is a legal 655
- RFC822 name identifying the set of mail addresses hosted by the med.example.com mail server. 656
- 657 And the XPath/XPointer value //xacml-context:Request/xacml-context:Resource/xacml-
- 658 context:ResourceContent/md:record/md:patient/ is a legal XPath/XPointer value identifying a
- 659 node-set in an XML document.
- 660 The question arises: how should a name that identifies a set of **subjects** or **resources** be
- 661 interpreted by the **PDP**, whether it appears in a **policy** or a request **context**? Are they intended to
- 662 represent just the node explicitly identified by the name, or are they intended to represent the entire
- 663 sub-tree subordinate to that node?
- In the case of *subjects*, there is no real entity that corresponds to such a node. So, names of this 664
- type always refer to the set of *subjects* subordinate in the name structure to the identified node. 665
- 666 Consequently, non-leaf **subject** names should not be used in equality functions, only in match
- 667 functions, such as "urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match" not
- 668 "urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal" (see Appendix A).

3.3.1.2. Effect

The *effect* of the *rule* indicates the rule-writer's intended consequence of a "True" evaluation for the *rule*. Two values are allowed: "Permit" and "Deny".

3.3.1.3. Condition

673 **Condition** represents a Boolean expression that refines the applicability of the **rule** beyond the **predicates** implied by its **target**. Therefore, it may be absent.

675 **3.3.2 Policy**

- From the data-flow model one can see that *rules* are not exchanged amongst system entities.

 Therefore, a *PAP* combines *rules* in a *policy*. A *policy* comprises four main components:
- 678 a *target*;
- a *rule-combining algorithm*-identifier;
- 680 a set of *rules*; and
- 681 obligations.

685

686 687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

Rules are described above. The remaining components are described in the following subsections.

684 **3.3.2.1.** Policy target

An XACML <PolicySet>, <Policy> or <Rule> element contains a <Target> element that specifies the set of *subjects*, *resources*, *actions* and *environments* to which it applies. The <Target> of a <PolicySet> or <Policy> may be declared by the writer of the <PolicySet> or <Policy>, or it may be calculated from the <Target> elements of the <PolicySet>, <Policy> and <Rule> elements that it contains.

A system entity that calculates a <Target> in this way is not defined by XACML, but there are two logical methods that might be used. In one method, the <Target> element of the outer <PolicySet> or <Policy> (the "outer component") is calculated as the *union* of all the <Target> elements of the referenced <PolicySet>, <Policy> or <Rule> elements (the "inner components"). In another method, the <Target> element of the outer component is calculated as the *intersection* of all the <Target> elements of the inner components. The results of evaluation in each case will be very different: in the first case, the <Target> element of the outer component makes it applicable to any *decision request* that matches the <Target> element of at least one inner component; in the second case, the <Target> element of the outer component makes it applicable only to *decision requests* that match the <Target> elements of every inner component. Note that computing the intersection of a set of <Target> elements is likely only practical if the target data-model is relatively simple.

In cases where the <Target> of a <Policy> is declared by the **policy** writer, any component <Rule> elements in the <Policy> that have the same <Target> element as the <Policy> element may omit the <Target> element. Such <Rule> elements inherit the <Target> of the <Policy> in which they are contained.

3.3.2.2. Rule-combining algorithm 706 707 The *rule-combining algorithm* specifies the procedure by which the results of evaluating the component rules are combined when evaluating the policy, i.e. the Decision value placed in the 708 709 response *context* by the *PDP* is the value of the *policy*, as defined by the *rule-combining* 710 algorithm. A policy may have combining parameters that affect the operation of the rule-711 combining algorithm. 712 See Appendix C for definitions of the normative *rule-combining algorithms*. 713 3.3.2.3. **Obligations** 714 **Obligations** may be added by the writer of the **policy**. 715 When a **PDP** evaluates a **policy** containing **obligations**, it returns certain of those **obligations** to 716 the **PEP** in the response **context**. Section 7.14 explains which **obligations** are to be returned. 3.3.3 Policy set 717 718 A *policy set* comprises four main components: 719 a target; 720 a policy-combining algorithm-identifier 721 a set of policies; and 722 obligations. 723 The *target* and *policy* components are described above. The other components are described in the following sub-sections. 724 3.3.3.1. Policy-combining algorithm 725 726 The *policy-combining algorithm* specifies the procedure by which the results of evaluating the component policies are combined when evaluating the policy set, i.e. the Decision value placed 727 in the response *context* by the *PDP* is the result of evaluating the *policy set*, as defined by the 728 729 policy-combining algorithm. A policy set may have combining parameters that affect the 730 operation of the *policy-combining algorithm*. 731 See Appendix C for definitions of the normative *policy-combining algorithms*. 3.3.3.2. **Obligations** 732 733 The writer of a *policy set* may add *obligations* to the *policy set*, in addition to those contained in the component policies and policy sets. 734 735 When a **PDP** evaluates a **policy set** containing **obligations**, it returns certain of those **obligations** 736 to the **PEP** in its response **context**. Section 7.14 explains which **obligations** are to be returned.

4. Examples (non-normative)

- 738 This section contains two examples of the use of XACML for illustrative purposes. The first example
- 739 is a relatively simple one to illustrate the use of *target*, *context*, matching functions and *subject*
- 740 *attributes*. The second example additionally illustrates the use of the *rule-combining algorithm*,
- 741 *conditions* and *obligations*.

737

742

743

4.1. Example one

4.1.1 Example policy

- Assume that a corporation named Medi Corp (identified by its domain name: med.example.com)
- has an **access control policy** that states, in English:
- Any user with an e-mail name in the "med.example.com" namespace is allowed to perform any *action* on any *resource*.
- An XACML *policy* consists of header information, an optional text description of the policy, a target, one or more rules and an optional set of **obligations**.

```
750
      [a02] <?xml version="1.0" encoding="UTF-8"?>
751
      [a03] <Policy
752
      [a04] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
753
      [a05] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
754
      [a06] xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
755
      http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
756
      [a07] PolicyId="urn:oasis:names:tc:example:SimplePolicy1"
757
      [a08] RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">
758
      [a09]
             <Description>
759
      [a10]
              Medi Corp access control policy
760
      [a11]
             </Description>
761
      [a12]
             <Target/>
762
      [a13]
             <Rule
763
      [a14] RuleId= "urn:oasis:names:tc:xacml:2.0:example:SimpleRule1"
764
      [a15]
             Effect="Permit">
765
      [a16]
              <Description>
766
      [a17]
               Any subject with an e-mail name in the med.example.com domain
767
      [a18]
               can perform any action on any resource.
768
      [a19]
              </Description>
769
      [a20] <Target>
770
      [a21]
              <Subjects>
771
      [a22]
               <Subject>
772
      [a23]
                 <SubjectMatch
773
                MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
      [a24]
774
      [a25]
                   <AttributeValue
775
      [a26]
                  DataType="http://www.w3.org/2001/XMLSchema#string">
776
      [a27]
                   med.example.com
777
      [a28]
                   </AttributeValue>
778
      [a29]
                   <SubjectAttributeDesignator</pre>
779
      [a30]
                   AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
780
      [a31]
                  DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
781
      [a32]
                  </SubjectMatch>
782
      [a33]
                 </Subject>
783
      [a34]
                </Subjects>
784
      [a35]
               </Target>
785
      [a36] </Rule>
786
      [a37] </Policy>
```

- 787 [a02] is a standard XML document tag indicating which version of XML is being used and what the
- 788 character encoding is.
- 789 [a03] introduces the XACML Policy itself.
- 790 [a04] [a05] are XML namespace declarations.
- 791 [a04] gives a URN for the XACML *policies* schema.
- 792 [a07] assigns a name to this *policy* instance. The name of a *policy* has to be unique for a given
- 793 **PDP** so that there is no ambiguity if one **policy** is referenced from another **policy**. The version
- attribute is omitted, so it takes its default value of "1.0".
- 795 [a08] specifies the algorithm that will be used to resolve the results of the various *rules* that may be
- 796 in the *policy*. The *deny-overrides rule-combining algorithm* specified here says that, if any *rule*
- evaluates to "Deny", then the *policy* must return "Deny". If all *rules* evaluate to "Permit", then the
- 798 *policy* must return "Permit". The *rule-combining algorithm*, which is fully described in Appendix
- 799 C, also says what to do if an error were to occur when evaluating any *rule*, and what to do with
- 800 *rules* that do not apply to a particular *decision request*.
- 801 [a09] [a11] provide a text description of the policy. This description is optional.
- 802 [a12] describes the decision requests to which this policy applies. If the subject, resource,
- 803 action and environment in a decision request do not match the values specified in the policy
- 804 *target*, then the remainder of the *policy* does not need to be evaluated. This *target* section is
- useful for creating an index to a set of *policies*. In this simple example, the *target* section says the
- 806 *policy* is applicable to any *decision request*.
- [a13] introduces the one and only *rule* in this simple *policy*.
- 808 [a14] specifies the identifier for this *rule*. Just as for a *policy*, each *rule* must have a unique
- identifier (at least unique for any **PDP** that will be using the **policy**).
- 810 [a15] says what **effect** this **rule** has if the **rule** evaluates to "True". **Rules** can have an **effect** of
- either "Permit" or "Deny". In this case, if the *rule* is satisfied, it will evaluate to "Permit", meaning
- that, as far as this one *rule* is concerned, the requested *access* should be permitted. If a *rule*
- evaluates to "False", then it returns a result of "NotApplicable". If an error occurs when evaluating
- the *rule*, then the *rule* returns a result of "Indeterminate". As mentioned above, the *rule*-
- 815 combining algorithm for the policy specifies how various rule values are combined into a single
- 816 *policy* value.
- 817 [a16] [a19] provide a text description of this *rule*. This description is optional.
- 818 [a20] introduces the *target* of the *rule*. As described above for the *target* of a policy, the *target* of
- 819 a *rule* describes the *decision requests* to which this *rule* applies. If the *subject*, *resource*,
- 820 **action** and **environment** in a **decision request** do not match the values specified in the **rule**
- 821 target, then the remainder of the rule does not need to be evaluated, and a value of
- 822 "NotApplicable" is returned to the *rule* evaluation.
- 823 The *rule target* is similar to the *target* of the *policy* itself, but with one important difference. [a23]-
- 824 [a32] spells out a specific value that the *subject* in the *decision request* must match. The
- 825 <SubjectMatch> element specifies a matching function in the MatchId attribute, a literal value of
- 826 "med.example.com" and a pointer to a specific *subject attribute* in the request *context* by means
- 827 of the <SubjectAttributeDesignator> element. The matching function will be used to
- 828 compare the literal value with the value of the *subject attribute*. Only if the match returns "True"
- will this *rule* apply to a particular *decision request*. If the match returns "False", then this *rule* will
- return a value of "NotApplicable".

- 831 [a36] closes the *rule*. In this *rule*, all the *work* is done in the <Target> element. In more complex
- 832 rules, the <Target> may have been followed by a <Condition> element (which could also be a
- set of *conditions* to be *AND*ed or *OR*ed together).
- 834 [a37] closes the *policy*. As mentioned above, this *policy* has only one *rule*, but more complex
- 835 *policies* may have any number of *rules*.

837

838 839

842

843

4.1.2 Example request context

Let's examine a hypothetical **decision request** that might be submitted to a **PDP** that executes the **policy** above. In English, the **access** request that generates the **decision request** may be stated as follows:

Bart Simpson, with e-mail name "bs@simpsons.com", wants to read his medical record at Medi Corp.

In XACML, the information in the *decision request* is formatted into a *request context* statement that looks as follows:

```
844
      [a38] <?xml version="1.0" encoding="UTF-8"?>
845
      [a39] <Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
846
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
847
      [a40] xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
848
      http://docs.oasis-open.org/xacml/access_control-xacml-2.0-context-schema-os.xsd">
849
      [a41] <Subject>
850
      [a42]
              <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"</pre>
851
      DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name">
852
      [a43] <AttributeValue>
853
      [a44]
               bs@simpsons.com
              </AttributeValue>
854
      [a45]
      [a46] </Attribute>
855
856
      [a47] </Subject>
857
      [a48] <Resource>
858
      [a49]
              <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-</pre>
859
      id" DataType="http://www.w3.org/2001/XMLSchema#anyURI">
860
              <AttributeValue>
      [a50]
861
      [a51]
               file://example/med/record/patient/BartSimpson
      [a52]
862
               </AttributeValue>
863
      [a53] </Attribute>
864
      [a54] </Resource>
865
      [a55] <Action>
866
      [a56]
              <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"</pre>
867
      DataType="http://www.w3.org/2001/XMLSchema#string">
868
             <AttributeValue>
      [a57]
869
      [a58]
                read
870
               </AttributeValue>
      [a59]
871
      [a60]
              </Attribute>
872
      [a61] </Action>
873
      [a62] <Environment/>
874
      [a63] </Request>
```

- [a38] [a40] contain the header information for the *request context*, and are used the same way as the header for the *policy* explained above.
- 877 The <Subject> element contains one or more attributes of the entity making the access request.
- There can be multiple *subjects*, and each *subject* can have multiple *attributes*. In this case, in
- 879 [a41] [a47], there is only one **subject**, and the **subject** has only one **attribute**: the **subject's**
- identity, expressed as an e-mail name, is "bs@simpsons.com". In this example, the subject-
- 881 category attribute is omitted. Therefore, it adopts its default value of "access-subject".

- The <Resource > element contains one or more attributes of the resource to which the subject (or
- 883 **subjects**) has requested **access**. There can be only one <Resource> per **decision request**¹.
- Lines [a48] [a54] contain the one *attribute* of the *resource* to which Bart Simpson has requested
- 885 access: the resource identified by its file URI, which is
- 886 "file://medico/record/patient/BartSimpson".
- 887 The <Action> element contains one or more attributes of the action that the subject (or
- 888 **subjects**) wishes to take on the **resource**. There can be only one **action** per **decision request**.
- 889 [a55] [a61] describe the identity of the action Bart Simpson wishes to take, which is "read".
- 890 The <Environment> element, [a62], is empty.
- 891 [a63] closes the *request context*. A more complex *request context* may have contained some
- 892 **attributes** not associated with the **subject**, the **resource** or the **action**. These would have been
- 893 placed in an optional <Environment> element following the <Action> element.
- The *PDP* processing this request *context* locates the *policy* in its policy repository. It compares
- 895 the *subject*, *resource*, *action* and *environment* in the request *context* with the *subjects*,
- 896 resources, actions and environments in the policy target. Since the policy target is empty, the
- 897 *policy* matches this *context*.
- 898 The **PDP** now compares the **subject**, **resource**, **action** and **environment** in the request **context**
- with the *target* of the one *rule* in this *policy*. The requested *resource* matches the <Target>
- 900 element and the requested action matches the <Target> element, but the requesting subject-id
- 901 attribute does not match "med.example.com".

4.1.3 Example response context

As a result of evaluating the policy, there is no *rule* in this *policy* that returns a "Permit" result for this request. The *rule-combining algorithm* for the *policy* specifies that, in this case, a result of "NotApplicable" should be returned. The response *context* looks as follows:

```
906 [a64] <?xml version="1.0" encoding="UTF-8"?>
```

- 907 [a65] <Response xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
- 908 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
- 909 xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:os
- 910 http://docs.oasis-open.org/xacml/xacml-core-2.0-context-schema-os.xsd">
- **911** [a66] <Result>
- **913** [a68] </Result>
- **914** [a69] </Response>
- 915 [a64] [a65] contain the same sort of header information for the response as was described above
- 916 for a *policy*.

902

903

904

- 917 The <Result> element in lines [a66] [a68] contains the result of evaluating the *decision request*
- 918 against the *policy*. In this case, the result is "NotApplicable". A *policy* can return "Permit", "Deny",
- 919 "NotApplicable" or "Indeterminate". Therefore, the **PEP** is required to deny **access**.
- 920 [a69] closes the response *context*.

¹ Some exceptions are described in the XACML Profile for Multiple Resources [MULT].

4.2. Example two

921

925

929

922 This section contains an example XML document, an example request context and example

923 XACML rules. The XML document is a medical record. Four separate rules are defined. These

924 illustrate a rule-combining algorithm, conditions and obligations.

4.2.1 Example medical record instance

926 The following is an instance of a medical record to which the example XACML *rules* can be 927 applied. The cord> schema is defined in the registered namespace administered by Medi 928

```
[a70] <?xml version="1.0" encoding="UTF-8"?>
930
      [a71] <record xmlns="urn:example:med:schemas:record"
931
      [a72] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
932
      [a73] <patient>
933
      [a74]
              <patientName>
934
      [a75]
               <first>Bartholomew</first>
935
      [a76]
               <last>Simpson</last>
     936
937
938
939
940
     [a81]
              <state>MA</state>
941
     [a82]
              <zip>12345</zip>
942
     [a83]
              <phone>555.123.4567</phone>
943
     [a84]
              <fax/>
     [a85]
[a86]
944
               <email/>
945
              </patientContact>
946
      [a87]
              <patientDoB>1992-03-21</patientDoB>
      [a88]
947
              <patientGender>male</patientGender>
948
      [a89]
              <patient-number>555555</patient-number>
     [a90]
949
              </patient>
950
      [a91] cardian>
              <parentGuardianId>HS001</parentGuardianId>
951
     [a92]
952
              <parentGuardianName>
     [a93]
953
     [a94]
               <first>Homer</first>
954
     [a95]
              <last>Simpson</last>
955
     [a96] </parentGuardianName>
956
     [a97]
             <parentGuardianContact>
957
     [a98]
              <street>27 Shelbyville Road</street>
958
     [a99]
              <city>Springfield</city>
959
     [a100]
              <state>MA</state>
960
      [a101] <zip>12345</zip>
961
               <phone>555.123.4567</phone>
      [a102]
962
      [a103]
               <fax/>
963
      [a104]
               <email>homers@aol.com</email>
964
      [a105]
              </parentGuardianContact>
965
            </parentGuardian>
      [a106]
966
     [a107] <primaryCarePhysician>
967
     [a108]
              <physicianName>
968
     [a109]
               <first>Julius</first>
969
     [a110]
              <last>Hibbert</last>
970
     [all1] </physicianName>
971
     [a112] <physicianContact>
972
     [a113]
              <street>1 First St</street>
973
     [a114]
              <city>Springfield</city>
974
     [a115]
              <state>MA</state>
975
      [a116]
               <zip>12345</zip>
976
      [a117]
               <phone>555.123.9012</phone>
977
      [a118]
                <fax>555.123.9013</fax>
```

```
978
       [a119]
                 <email/>
 979
       [a120]
              </physicianContact>
 980
       [a121]
                <registrationID>ABC123</registrationID>
 981
       [a122] </primaryCarePhysician>
 982
       [a123] <insurer>
 983
       [a124]
                 <name>Blue Cross</name>
 984
       [a125]
                 <street>1234 Main St</street>
 985
       [a126]
                 <city>Springfield</city>
 986
       [a127]
                 <state>MA</state>
 987
       [a128]
                 <zip>12345</zip>
 988
       [a129]
                 <phone>555.123.5678</phone>
 989
       [a130]
                 <fax>555.123.5679</fax>
 990
       [a131]
                 <email/>
 991
       [a132]
               </insurer>
 992
       [a133] <medical>
 993
       [a134]
               <treatment>
 994
       [a135]
                <drug>
 995
       [a136]
                 <name>methylphenidate hydrochloride
 996
       [a137]
                 <dailyDosage>30mgs</dailyDosage>
 997
       [a138]
                  <startDate>1999-01-12</startDate>
 998
       [a139]
                 </drug>
 999
       [a140]
                 <comment>
1000
       [a141]
                  patient exhibits side-effects of skin coloration and carpal
1001
       degeneration
1002
       [a142]
                 </comment>
1003
       [a143]
                </treatment>
1004
       [a144]
                <result>
1005
       [a145]
                <test>blood pressure</test>
1006
       [a146]
                <value>120/80</value>
1007
       [a147]
                <date>2001-06-09</date>
1008
       [a148]
                <performedBy>Nurse Betty</performedBy>
1009
       [a149]
               </result>
1010
       [a150] </medical>
1011
       [a151] </record>
```

4.2.2 Example request context

The following example illustrates a request *context* to which the example *rules* may be applicable.

It represents a request by the physician Julius Hibbert to read the patient date of birth in the record of Bartholomew Simpson.

```
1016
       [a152] <?xml version="1.0" encoding="UTF-8"?>
1017
       [a153] <Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
1018
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
1019
       urn:oasis:names:tc:xacml:2.0:context:schema:os http://docs.oasis-
1020
       open.org/xacml/access_control-xacml-2.0-context-schema-os.xsd">
1021
       [a154] <Subject>
1022
       [a155]
               <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject-category"</pre>
1023
       DataType="http://www.w3.org/2001/XMLSchema#anyURI">
1024
                <AttributeValue>urn:oasis:names:tc:xacml:1.0:subject-category:access-
1025
       subject</AttributeValue>
1026
       [a157]
               </Attribute>
1027
       [a158] <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
1028
       DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="med.example.com">
1029
       [a159]
               <AttributeValue>CN=Julius Hibbert</AttributeValue>
1030
       [a160] </Attribute>
1031
       [a161] <a href="deltatributeId="urn:oasis:names:tc:xacml:1.0:subject:name-
1032
       format DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1033
       Issuer="med.example.com">
1034
       [a162]
                <AttributeValue>
1035
       [a163]
                 urn:oasis:names:tc:xacml:1.0:datatype:x500name
1036
       [a164]
                 </AttributeValue>
```

```
1037
       [a165] </Attribute>
1038
       [a166] <Attribute
1039
       AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
1040
       DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="med.example.com">
1041
               <AttributeValue>physician</AttributeValue>
1042
        [a168] </Attribute>
1043
        [a169] <Attribute
1044
       AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:physician-id"
1045
       DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="med.example.com">
1046
        [a170]
                 <AttributeValue>jh1234</AttributeValue>
1047
        [a171]
                </Attribute>
1048
        [a172] </Subject>
1049
        [a173] <Resource>
1050
       [a174]
               <ResourceContent>
1051
       [a175]
                 <md:record xmlns:md="urn:example:med:schemas:record"
1052
       xsi:schemaLocation="urn:example:med:schemas:record
1053
       http:www.med.example.com/schemas/record.xsd">
1054
        [a176] <md:patient>
1055
                  <md:patientDoB>1992-03-21</md:patientDoB>
        [a177]
1056
        [a178]
                   <md:patient-number>555555</md:patient-number>
        [a179]
1057
                  </md:patient>
1058
        [a180]
                 </md:record>
       [a181] </ResourceContent>
[a182] <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-</pre>
1059
1060
1061
        id" DataType="http://www.w3.org/2001/XMLSchema#string">
1062
        [a183]
                  <AttributeValue>
1063
        [a184]
                  //med.example.com/records/bart-simpson.xml#
1064
        [a185] xmlns(md=:Resource/ResourceContent/xpointer
1065
        [a186] (/md:record/md:patient/md:patientDoB)
1066
                 </AttributeValue>
1067
       [a188] </Attribute>
1068
       [a189] </Resource>
1069
       [a190] <Action>
1070
        [a191] <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1071
       DataType="http://www.w3.org/2001/XMLSchema#string">
1072
        [a192] <a href="AttributeValue">AttributeValue</a>
        [a193] </Attribute>
[a194] </Action>
1073
1074
        [a195] <Environment/>
1075
1076
        [a196] </Request>
1077
       [a152] - [a153] Standard namespace declarations.
1078
        [a154] - [a172] Subject attributes are placed in the <Subject> element of the <Request>
1079
        element. Each attribute consists of the attribute meta-data and the attribute value. There is only
1080
        one subject involved in this request.
1081
        [a155] - [a157] Each <Subject> element has a SubjectCategory attribute. The value of this
1082
        attribute describes the role that the related subject plays in making the decision request. The
1083
        value of "access-subject" denotes the identity for which the request was issued.
1084
       [a158] - [a160] Subject subject-id attribute.
1085
        [a161] - [a165] The format of the subject-id.
1086
       [a166] - [a168] Subject role attribute.
1087
       [a169] - [a171] Subject physician-id attribute.
1088
        [a173] - [a189] Resource attributes are placed in the <Resource> element of the <Request>
1089
        element. Each attribute consists of attribute meta-data and an attribute value.
```

1090 [a174] - [a181] Resource content. The XML resource instance, access to all or part of which may 1091 be requested, is placed here. 1092 [a182] - [a188] The identifier of the Resource instance for which access is requested, which is an 1093 XPath expression into the <ResourceContent> element that selects the data to be accessed. 1094 [a190] - [a194] Action attributes are placed in the <Action> element of the <Request> element. 1095 [a192] Action identifier. 1096 [a195] The empty <Environment> element. 4.2.3 Example plain-language rules 1097 1098 The following plain-language rules are to be enforced: 1099 Rule 1: A person, identified by his or her patient number, may read any record for which he 1100 or she is the designated patient. 1101 Rule 2: A person may read any record for which he or she is the designated parent or 1102 guardian, and for which the patient is under 16 years of age. 1103 Rule 3: A physician may write to any medical element for which he or she is the designated primary care physician, provided an email is sent to the patient. 1104 1105 Rule 4: An administrator shall not be permitted to read or write to medical elements of a 1106 patient record. 1107 These *rules* may be written by different *PAP*s operating independently, or by a single *PAP*. 4.2.4 Example XACML rule instances 1108 1109 4.2.4.1. Rule 1 1110 Rule 1 illustrates a simple rule with a single <Condition> element. It also illustrates the use of 1111 the <VariableDefinition> element to define a function that may be used throughout the 1112 **policy**. The following XACML <Rule> instance expresses Rule 1: 1113 [a197] <?xml version="1.0" encoding="UTF-8"?> 1114 [a198] <Policy 1115 [a199] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" xmlns:xacml-1116 context="urn:oasis:names:tc:xacml:2.0:context:schema:os" 1117 [a200] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=" 1118 urn:oasis:names:tc:xacml:2.0:policy:schema:os http://docs.oasis-1119 open.org/xacml/access control-xacml-2.0-context-schema-os.xsd" 1120 [a201] xmlns:md="http://www.med.example.com/schemas/record.xsd" 1121 [a202] PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:1" 1122 [a203] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-1123 algorithm:deny-overrides"> 1124 [a204] <PolicyDefaults> 1125 [a205] <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-1126 19991116</XPathVersion> 1127 [a206] </PolicyDefaults> 1128 [a207] <Target/> 1129 [a208] <VariableDefinition VariableId="17590034"> 1130 [a209] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> 1131 [a210] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-</pre>

and-only">

```
1133
       [a211]
                  <SubjectAttributeDesignator
1134
       AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:patient-number"
1135
                DataType="http://www.w3.org/2001/XMLSchema#string"/>
1136
       [a213]
              </Apply>
1137
       [a214]
                <Apply
1138
       [a215] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1139
       [a216]
                 <AttributeSelector
1140
       [a217]
                 RequestContextPath="//xacml-context:Resource/xacml-
1141
       context:ResourceContent/md:record/md:patient/md:patient-number/text()"
1142
       [a218]
                DataType="http://www.w3.org/2001/XMLSchema#string"/>
1143
       [a219]
                </Apply>
1144
       [a220] </Apply>
1145
       [a221] </VariableDefinition>
1146
       [a222] <Rule
1147
       [a223] RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:1"
1148
      [a224] Effect="Permit">
      [a225] <Description>
1149
1150
      [a226]
                 A person may read any medical record in the
1151
      [a227]
                 http://www.med.example.com/schemas/record.xsd namespace
1152
       [a228]
                for which he or she is the designated patient
1153
       [a229] </Description>
1154
       [a230] <Target>
1155
       [a231]
               <Resources>
1156
       [a232]
                 <Resource>
1157
       [a233]
                   <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-</pre>
1158
       equal">
1159
                  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
       [a234]
1160
       [a235]
                     urn:example:med:schemas:record
1161
                   </AttributeValue>
       [a236]
1162
       [a237]
                   <ResourceAttributeDesignator AttributeId=</pre>
1163
       [a238]
                   "urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1164
       [a239]
                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
1165
       [a240]
                  </ResourceMatch>
1166
       [a241]
                  <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-</pre>
1167
       node-match">
1168
       [a242] <a href="http://www.w3.org/2001/XMLSchema#string">
1169
       [a243]
                     /md:record
1170
       [a244]
                   </AttributeValue>
1171
       [a245]
                   <ResourceAttributeDesignator
1172
       AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
1173
       [a246]
                    DataType="http://www.w3.org/2001/XMLSchema#string"/>
1174
       [a247]
                  </ResourceMatch>
1175
       [a248]
                 </Resource>
1176
      [a249]
               </Resources>
1177
       [a250] <Actions>
1178
      [a251]
                <Action>
1179
       [a252]
1180
       [a253]
                  MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1181
       [a254]
                   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1182
       [a255]
                      read
1183
       [a256]
                   </AttributeValue>
1184
       [a257]
                   <ActionAttributeDesignator
1185
       [a258]
                    AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1186
       [a259]
                    DataType="http://www.w3.org/2001/XMLSchema#string"/>
1187
       [a260]
                  </ActionMatch>
1188
                </Action>
       [a261]
1189
       [a262]
                </Actions>
1190
       [a263] </Target>
1191
       [a264]
              <Condition>
1192
       [a265]
                <VariableReference VariableId="17590034"/>
1193
       [a266]
              </Condition>
1194
       [a267] </Rule>
```

- 1195 [a268] </Policy>
- 1196 [a199] [a201] XML namespace declarations.
- 1197 [a205] XPath expressions in the **policy** are to be interpreted according to the 1.0 version of the
- 1198 XPath specification.
- 1199 [a208] [a221] A <VariableDefinition> element. It defines a function that evaluates the truth
- 1200 of the statement: the patient-number *subject attribute* is equal to the patient-number in the
- 1201 *resource*.
- 1202 [a209] The FunctionId attribute names the function to be used for comparison. In this case,
- 1203 comparison is done with the "urn:oasis:names:tc:xacml:1.0:function:string-equal" function; this
- 1204 function takes two arguments of type "http://www.w3.org/2001/XMLSchema#string".
- 1205 [a210] The first argument of the variable definition is a function specified by the FunctionId
- 1206 attribute. Since urn:oasis:names:tc:xacml:1.0:function:string-equal takes
- arguments of type "http://www.w3.org/2001/XMLSchema#string" and
- 1208 SubjectAttributeDesignator selects a bag of type
- 1209 "http://www.w3.org/2001/XMLSchema#string", "urn:oasis:names:tc:xacml:1.0:function:string-one-
- 1210 and-only" is used. This function guarantees that its argument evaluates to a *bag* containing exactly
- 1211 one value.
- 1212 [a211] The SubjectAttributeDesignator selects a bag of values for the patient-number
- 1213 *subject attribute* in the request *context*.
- 1214 [a215] The second argument of the variable definition is a function specified by the FunctionId
- 1215 attribute. Since "urn:oasis:names:tc:xacml:1.0:function:string-equal" takes arguments of type
- 1216 "http://www.w3.org/2001/XMLSchema#string" and the AttributeSelector selects a bag of type
- 1217 "http://www.w3.org/2001/XMLSchema#string", "urn:oasis:names:tc:xacml:1.0:function:string-one-
- 1218 and-only" is used. This function guarantees that its argument evaluates to a *bag* containing exactly
- 1219 one value.
- 1220 [a216] The <attributeSelector> element selects a bag of values from the request context
- using a free-form XPath expression. In this case, it selects the value of the patient-number in
- the *resource*. Note that the namespace prefixes in the XPath expression are resolved with the
- 1223 standard XML namespace declarations.
- 1224 [a223] Rule identifier.
- 1225 [a224] Rule effect declaration. When a rule evaluates to 'True' it emits the value of the Effect
- 1226 attribute. This value is then combined with the Effect values of other *rules* according to the *rule-*
- 1227 **combining algorithm**.
- 1228 [a225] [a229] Free form description of the *rule*.
- 1229 [a230] [a263] A *rule target* defines a set of *decision requests* that the *rule* is intended to
- 1230 evaluate. In this example, the <Subjects> and <Environments> elements are omitted.
- 1231 [a231] [a249] The <Resources> element contains a disjunctive sequence of <Resource>
- 1232 elements. In this example, there is just one.
- 1233 [a232] [a248] The <Resource> element encloses the conjunctive sequence of
- 1234 ResourceMatch elements. In this example, there are two.
- 1235 [a233] [a240] The first <ResourceMatch> element compares its first and second child elements
- 1236 according to the matching function. A match is positive if the value of the first argument matches
- any of the values selected by the second argument. This match compares the target namespace of
- the requested document with the value of "urn:example:med:schemas:record".

- 1239 [a233] The MatchId attribute names the matching function.
- 1240 [a235] Literal attribute value to match.
- 1241 [a237] - [a239] The <ResourceAttributeDesignator> element selects the target namespace
- from the resource contained in the request context. The attribute name is specified by the 1242
- 1243 AttributeId.
- 1244 [a241] - [a247] The second <ResourceMatch > element. This match compares the results of two
- 1245 XPath expressions. The second XPath expression is the location path to the requested XML
- 1246 element and the first XPath expression is the literal value "/md:record". The "xpath-node-match"
- 1247 function evaluates to "True" if the requested XML element is below the "/md:record" element.
- 1248 [a250] - [a262] The <Actions> element contains a disjunctive sequence of <Action> elements.
- 1249 In this case, there is just one <Action> element.
- 1250 [a251] - [a261] The <Action> element contains a conjunctive sequence of <ActionMatch>
- 1251 elements. In this case, there is just one <actionMatch> element.
- 1252 [a252] - [a260] The <actionMatch> element compares its first and second child elements
- 1253 according to the matching function. The match is positive if the value of the first argument matches
- any of the values selected by the second argument. In this case, the value of the action-id 1254
- 1255 action attribute in the request context is compared with the literal value "read".
- 1256 [a264] - [a266] The <Condition> element. A condition must evaluate to "True" for the rule to be
- 1257 applicable. This *condition* contains a reference to a variable definition defined elsewhere in the
- 1258 policy.

4.2.4.2. Rule 2 1259

- 1260 Rule 2 illustrates the use of a mathematical function, i.e. the <apply> element with functionId
- 1261 "urn:oasis:names:tc:xacml:1.0:function:date-add-vearMonthDuration" to calculate the date of the
- 1262 patient's sixteenth birthday. It also illustrates the use of predicate expressions, with the
- 1263 functionId "urn:oasis:names:tc:xacml:1.0:function:and". This example has one function
- embedded in the <Condition> element and another one referenced in a 1264
- 1265 <VariableDefinition> element.
- 1266 [a269] <?xml version="1.0" encoding="UTF-8"?>
- 1267 [a270] < Policy
- 1268 [a271] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" xmlns:xacml-
- 1269 context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
- 1270 [a272] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
- 1271 xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
- 1272 http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
- 1273 [a273] xmlns:xf="http://www.w3.org/TR/2002/WD-xquery-operators-20020816/#"
- 1274 [a274] xmlns:md="http:www.med.example.com/schemas/record.xsd"
- 1275 [a275] PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:2"
- 1276 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-1277 overrides">
- 1278 [a276] <PolicyDefaults>
- 1279 <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-[a277]
- 1280 19991116</XPathVersion>
- 1281 [a278] </PolicyDefaults>
- 1282
- [a279] <Target/>
 [a280] <VariableDefinition VariableId="17590035"> 1283
- 1284 [a281] <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:date-less-or-</pre>
- 1285 equal">
- 1286 [a282] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-</pre>
- 1287 only">

```
1288
      [a283]
                 <EnvironmentAttributeDesignator</pre>
1289
      [a284]
                AttributeId= "urn:oasis:names:tc:xacml:1.0:environment:current-date"
1290
       [a285]
                DataType="http://www.w3.org/2001/XMLSchema#date"/>
1291
       [a286] </Apply>
1292
      [a287]
                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-add-</pre>
1293
       yearMonthDuration">
1294
       [a288]
                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-</pre>
1295
       only">
1296
       [a289]
                   <AttributeSelector RequestContextPath=</pre>
1297
       [a290]
                   "//md:record/md:patient/md:patientDoB/text()"
1298
       [a291]
                  DataType="http://www.w3.org/2001/XMLSchema#date"/>
1299
       [a292]
                  </Apply>
1300
       [a293]
                  <AttributeValue
1301
                DataType="http://www.w3.org/TR/2002/WD-xquery-operators-
       [a294]
1302
       20020816#yearMonthDuration">
1303
                <xf:dt-yearMonthDuration>
       [a295]
1304
       [a296]
                  P16Y
1305
       [a297]
                  </xf:dt-yearMonthDuration>
1306
      [a298]
                 </AttributeValue>
1307
       [a299] </Apply>
1308
       [a300] </Apply>
1309
       [a301] </VariableDefinition>
       [a302] <Rule
[a303] RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:2"</pre>
1310
1311
       [a304] Effect="Permit">
1312
1313
       [a305]
              <Description>
1314
       [a306]
               A person may read any medical record in the
1315
       [a307]
               http://www.med.example.com/records.xsd namespace
1316
       [a308]
               for which he or she is the designated parent or guardian,
1317
       [a309]
               and for which the patient is under 16 years of age
1318
      [a310] </Description>
1319
      [a311] <Target>
1320
      [a312] <Resources>
1321
      [a313]
                <Resource>
1322
      [a314]
                  <ResourceMatch
       [a315]
                 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1323
1324
       [a316]
                  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1325
       [a317]
                      http://www.med.example.com/schemas/record.xsd
1326
       [a318]
                    </AttributeValue>
1327
       [a319]
                   <ResourceAttributeDesignator AttributeId=</pre>
1328
       "urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1329
       [a320]
                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
1330
       [a321]
                  </ResourceMatch>
1331
       [a322]
                  <ResourceMatch
1332
       [a323]
                 MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
1333
      [a324]
                  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1334
      [a325]
                      /md:record
1335
                   </AttributeValue>
       [a326]
1336
                   <ResourceAttributeDesignator</pre>
       [a327]
1337
       AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
1338
       [a328] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1339
       [a329]
                  </ResourceMatch>
1340
       [a330]
                 </Resource>
1341
       [a331]
                </Resources>
1342
       [a332]
                <Actions>
1343
       [a333]
                <Action>
1344
       [a334]
                 <ActionMatch
1345
                 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       [a335]
1346
       [a336]
                  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1347
       [a337]
1348
       [a338]
                   </AttributeValue>
```

```
1349
       [a339]
                     <ActionAttributeDesignator
1350
       AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1351
        [a340] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1352
        [a341]
                  </ActionMatch>
1353
        [a342]
                  </Action>
1354
        [a343] </Actions>
1355
        [a344] </Target>
1356
        [a345] <Condition>
1357
        [a346] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
1358
        [a347]
                 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1359
        [a348]
                   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-</pre>
1360
        and-only">
1361
        [a349]
                    <SubjectAttributeDesignator
1362
       AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:
1363
        [a350] parent-guardian-id"
1364
                   DataType="http://www.w3.org/2001/XMLSchema#string"/>
        [a351]
1365
        [a352]
                  </Apply>
1366
        [a353]
                 <Apply
1367
                  FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-
       [a354]
1368
       only">
1369
        [a355]
                    <AttributeSelector
                 RequestContextPath="//xacml-context:Resource/xacml-
1370
        [a356]
1371
        context:ResourceContent/md:record/md:parentGuardian/md:parentGuardianId/text()"
1372
        [a357] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1373
        [a358]
                    </Apply>
1374
                  </Apply>
        [a359]
1375
                  <VariableReference VariableId="17590035"/>
        [a360]
1376
        [a361]
                </Apply>
1377
        [a362] </Condition>
1378
        [a363] </Rule>
1379
        [a364] </Policy>
1380
        [a280] - [a301] The <VariableDefinition> element contains part of the condition (i.e. is the
1381
        patient under 16 years of age?). The patient is under 16 years of age if the current date is less than
        the date computed by adding 16 to the patient's date of birth.
1382
1383
        [a281] - [a300] "urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal" is used to compute the
1384
        difference of two date arguments.
1385
        [a282] - [a286] The first date argument uses "urn:oasis:names:tc:xacml:1.0:function:date-one-and-
1386
        only" to ensure that the bag of values selected by its argument contains exactly one value of type
1387
        "http://www.w3.org/2001/XMLSchema#date".
1388
        [a284] The current date is evaluated by selecting the
1389
        "urn:oasis:names:tc:xacml:1.0:environment:current-date" environment attribute.
1390
        [a287] - [a299] The second date argument uses "urn:oasis:names:tc:xacml:1.0:function:date-add-
1391
        yearMonthDuration" to compute the date of the patient's sixteenth birthday by adding 16 years to
1392
        the patient's date of birth. The first of its arguments is of type
        "http://www.w3.org/2001/XMLSchema#date" and the second is of type
1393
1394
        "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration".
1395
        [a289] The <attributeSelector> element selects the patient's date of birth by taking the XPath
1396
        expression over the resource content.
1397
        [a293] - [a298] Year Month Duration of 16 years.
1398
        [a311] - [a344] Rule declaration and rule target. See Rule 1 in Section 4.2.4.1 for the detailed
1399
        explanation of these elements.
1400
        [a345] - [a362] The <Condition> element. The condition must evaluate to "True" for the rule to
1401
        be applicable. This condition evaluates the truth of the statement: the requestor is the designated
```

1 February 2005

Page 35 of 141

access_control-xacml-2.0-core-spec-os

Copyright © OASIS Open 2004. All Rights Reserved.

- parent or guardian and the patient is under 16 years of age. It contains one embedded ">
- 1403 element and one referenced <VariableDefinition> element.
- 1404 [a346] The *condition* uses the "urn:oasis:names:tc:xacml:1.0:function:and" function. This is a
- 1405 Boolean function that takes one or more Boolean arguments (2 in this case) and performs the
- logical "AND" operation to compute the truth value of the expression.
- 1407 [a347] [a359] The first part of the *condition* is evaluated (i.e. is the requestor the designated
- parent or guardian?). The function is "urn:oasis:names:tc:xacml:1.0:function:string-equal" and it
- takes two arguments of type "http://www.w3.org/2001/XMLSchema#string".
- 1410 [a348] designates the first argument. Since "urn:oasis:names:tc:xacml:1.0:function:string-equal"
- takes arguments of type "http://www.w3.org/2001/XMLSchema#string",
- 1412 "urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" is used to ensure that the *subject*
- 1413 **attribute** "urn:oasis:names:tc:xacml:2.0:example:attribute:parent-guardian-id" in the request
- 1414 *context* contains exactly one value.
- 1415 [a353] designates the second argument. The value of the subject attribute
- 1416 "urn:oasis:names:tc:xacml:2.0:example:attribute:parent-guardian-id" is selected from the request
- 1417 *context* using the <SubjectAttributeDesignator> element.
- 1418 [a354] As above, the "urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" is used to ensure
- that the **bag** of values selected by it's argument contains exactly one value of type
- 1420 "http://www.w3.org/2001/XMLSchema#string".
- 1421 [a355] The second argument selects the value of the <md:parentGuardianId> element from the
- 1422 **resource** content using the <attributeSelector> element. This element contains a free-form
- 1423 XPath expression, pointing into the request *context*. Note that all namespace prefixes in the XPath
- 1424 expression are resolved with standard namespace declarations. The AttributeSelector
- evaluates to the *bag* of values of type "http://www.w3.org/2001/XMLSchema#string".
- 1426 [a360] references the <VariableDefinition> element, where the second part of the condition
- 1427 is defined.
- 1428 **4.2.4.3.** Rule 3
- 1429 Rule 3 illustrates the use of an *obligation*. The XACML <Rule> element syntax does not include
- 1430 an element suitable for carrying an *obligation*, therefore Rule 3 has to be formatted as a
- 1431 <Policy> element.
- 1432 [a365] <?xml version="1.0" encoding="UTF-8"?>
- 1433 [a366] <Policy
- 1434 [a367] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" xmlns:xacml-
- 1435 context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
- 1436 [a368] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
- 1437 [a369] xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
- 1438 http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
- 1439 [a370] xmlns:md="http:www.med.example.com/schemas/record.xsd"
- 1440 [a371] PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:3"
- 1441 [a372] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
- 1442 algorithm:deny-overrides">
- 1443 [a373] <Description>
- 1444 [a374] Policy for any medical record in the
- 1445 [a375] http://www.med.example.com/schemas/record.xsd namespace
- 1446 [a376] </Description>
- 1447 [a377] <PolicyDefaults>
- 1448 [a378] <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-
- 1449 19991116</XPathVersion>
- 1450 [a379] </PolicyDefaults>

```
1451
      [a380] <Target>
1452
       [a381]
              <Resources>
1453
       [a382]
               <Resource>
1454
       [a383]
                 <ResourceMatch
1455
      [a384]
                MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1456
       [a385]
                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1457
       [a386]
                  urn:example:med:schemas:record
1458
       [a387]
                  </AttributeValue>
1459
       [a388]
                  <ResourceAttributeDesignator AttributeId=</pre>
1460
       [a389]
                   "urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
1461
       [a390]
                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
1462
       [a391]
                 </ResourceMatch>
1463
       [a392]
                </Resource>
1464
       [a393] </Resources>
1465
       [a394] </Target>
1466
       [a395] <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:3"
1467
       [a396] Effect="Permit">
1468
       [a397] <Description>
1469
      [a398] A physician may write any medical element in a record
1470
      [a399]
               for which he or she is the designated primary care
       [a400]
1471
               physician, provided an email is sent to the patient
1472
       [a401]
               </Description>
1473
       [a402]
              <Target>
1474
       [a403]
                <Subjects>
1475
       [a404]
                 <Subject>
1476
       [a405]
                  <SubjectMatch
1477
                 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       [a406]
1478
       [a407]
                  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1479
       [a408]
                    physician
1480
       [a409]
                    </AttributeValue>
1481
                    <SubjectAttributeDesignator AttributeId=</pre>
       [a410]
1482
       "urn:oasis:names:tc:xacml:2.0:example:attribute:role"
1483
       [a411]
                   DataType="http://www.w3.org/2001/XMLSchema#string"/>
1484
      [a412]
                  </SubjectMatch>
1485
      [a413]
                </Subject>
1486
       [a414]
              </Subjects>
1487
       [a415]
               <Resources>
1488
       [a416]
                 <Resource>
1489
       [a417]
                  <ResourceMatch
1490
       [a418]
                 MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
1491
       [a419]
                   <AttributeValue
1492
       [a420]
                    DataType="http://www.w3.org/2001/XMLSchema#string">
1493
       [a421]
                      /md:record/md:medical
1494
      [a422]
                    </AttributeValue>
1495
       [a423]
                   <ResourceAttributeDesignator</pre>
1496
       AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
1497
       [a424]
                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
1498
       [a425]
                  </ResourceMatch>
1499
       [a426]
                 </Resource>
1500
       [a427]
                </Resources>
1501
               <Actions>
       [a428]
1502
       [a429]
                 <Action>
1503
       [a430]
                  <ActionMatch
1504
       [a431]
                  MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1505
       [a432]
                   <AttributeValue
1506
                   DataType="http://www.w3.org/2001/XMLSchema#string">
       [a433]
1507
       [a434]
                     write
1508
       [a435]
                   </AttributeValue>
1509
       [a436]
                   <ActionAttributeDesignator
1510
       AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1511
       [a437]
                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
1512
       [a438]
                  </ActionMatch>
```

```
1513
       [a439]
                 </Action>
1514
       [a440]
                </Actions>
1515
       [a441] </Target>
1516
       [a442] <Condition>
1517
       [a443] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1518
       [a444]
                 <Apply
1519
       [a445] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1520
       [a446]
                 <SubjectAttributeDesignator
1521
       [a447]
                  AttributeId="urn:oasis:names:tc:xacml:2.0:example:
1522
       attribute:physician-id"
1523
       [a448]
                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1524
       [a449]
                 </Apply>
1525
       [a450]
                 <Apply
1526
       [a451] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1527
       [a452]
                 <AttributeSelector RequestContextPath=</pre>
1528
       [a453]
                  "//xacml-context:Resource/xacml-
1529
       context:ResourceContent/md:record/md:primaryCarePhysician/md:registrationID/text(
1530
1531
       [a454]
                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1532
       [a455] </Apply>
1533
       [a456]
                 </Apply>
1534
       [a457] </Condition>
       [a458] </Rule>
[a459] <Obligations>
1535
1536
1537
       [a460]
               <0bligation
1538
       ObligationId="urn:oasis:names:tc:xacml:example:obligation:email"
1539
       [a461] FulfillOn="Permit">
1540
       [a462]
                <AttributeAssignment
1541
       AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:mailto"
1542
       [a463] DataType="http://www.w3.org/2001/XMLSchema#string">
1543
       [a464]
                <AttributeSelector RequestContextPath=
1544
       [a465]
                 "//md:/record/md:patient/md:patientContact/md:email"
1545
       [a466]
                DataType="http://www.w3.org/2001/XMLSchema#string"/>
1546
       [a467] </AttributeAssignment>
1547
                <a href="#">AttributeAssignment</a>
       [a468]
1548
       AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:text"
1549
       [a469] DataType="http://www.w3.org/2001/XMLSchema#string">
1550
       [a470]
                  Your medical record has been accessed by:
               </AttributeAssignment>
<AttributeAssignment
1551
       [a471]
1552
       [a472]
1553
       AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:text"
1554
       [a473] DataType="http://www.w3.org/2001/XMLSchema#string">
1555
       [a474]
                 <SubjectAttributeDesignator
1556
       AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
1557
       [a475]
                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1558
       [a476]
                 </AttributeAssignment>
1559
       [a477] </Obligation>
1560
       [a478] </Obligations>
1561
       [a479] </Policy>
1562
       [a366] - [a372] The <Policy> element includes standard namespace declarations as well as policy
1563
       specific parameters, such as PolicyId and RuleCombiningAlgId.
1564
       [a371] Policy identifier. This parameter allows the policy to be referenced by a policy set.
1565
       [a372] The Rule combining algorithm identifies the algorithm for combining the outcomes of rule
1566
       evaluation.
```

[a379] - [a394] Policy target. The policy target defines a set of applicable decision requests. The

structure of the <Target> element in the <Policy> is identical to the structure of the <Target>

1567

1568 1569 [a373] - [a376] Free-form description of the *policy*.

- 1570 element in the <Rule>. In this case, the *policy target* is the set of all XML resources that conform
- to the namespace "urn:example:med:schemas:record".
- 1572 [a395] The only <Rule> element included in this <Policy>. Two parameters are specified in the
- 1573 rule header: RuleId and Effect.
- 1574 [a402] [a441] The *rule target* further constrains the *policy target*.
- 1575 [a405] [a412] The <SubjectMatch> element targets the *rule* at *subjects* whose
- 1576 "urn:oasis:names:tc:xacml:2.0:example:attribute:role" *subject attribute* is equal to "physician".
- 1577 [a417] [a425] The <ResourceMatch> element targets the *rule* at *resources* that match the
- 1578 XPath expression "/md:record/md:medical".
- 1579 [a430] [a438] The <ActionMatch> element targets the *rule* at *actions* whose
- "urn:oasis:names:tc:xacml:1.0:action:action-id" action attribute is equal to "write".
- 1581 [a442] [a457] The <Condition> element. For the *rule* to be applicable to the *decision request*,
- 1582 the *condition* must evaluate to "True". This *condition* compares the value of the
- 1583 "urn:oasis:names:tc:xacml:2.0:example:attribute:physician-id" *subject attribute* with the value of
- 1584 the <registrationId> element in the medical record that is being accessed.
- 1585 [a459] [a478] The <Obligations > element. *Obligations* are a set of operations that must be
- performed by the *PEP* in conjunction with an *authorization decision*. An *obligation* may be
- 1587 associated with a "Permit" or "Deny" authorization decision. The element contains a single
- 1588 *obligation*.
- 1589 [a460] [a477] The <Obligation> element consists of the ObligationId attribute, the
- 1590 **authorization decision** value for which it must be fulfilled, and a set of **attribute** assignments. The
- 1591 **PDP** does not resolve the attribute assignments. This is the job of the **PEP**.
- 1592 [a460] The ObligationId attribute identifies the *obligation*. In this case, the *PEP* is required to
- 1593 send email.
- 1594 [a461] The Fulfillon attribute defines the *authorization decision* value for which this
- obligation must be fulfilled. In this case, when access is permitted.
- 1596 [a462] [a467] The first parameter indicates where the **PEP** will find the email address in the
- 1597 resource.
- 1598 [a468] [a471] The second parameter contains literal text for the email body.
- 1599 [a472] [a476] The third parameter indicates where the **PEP** will find further text for the email body
- 1600 in the resource.

1601 **4.2.4.4.** Rule 4

- 1602 Rule 4 illustrates the use of the "Deny" Effect value, and a <Rule> with no <Condition> 1603 element.
- 1001
- 1604 [a480] <?xml version="1.0" encoding="UTF-8"?>
- **1605** [a481] < Policy
- 1606 [a482] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
- 1607 [a483] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
- 1608 xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
- 1609 http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
- 1610 [a484] xmlns:md="http:www.med.example.com/schemas/record.xsd"
- 1611 [a485] PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:4"

```
1612
       [a486] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1613
       algorithm:deny-overrides">
1614
       [a487] <PolicyDefaults>
1615
                <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-
       [a488]
1616
       19991116</XPathVersion>
1617
       [a489] </PolicyDefaults>
1618
       [a490] <Target/>
1619
       [a491] <Rule
1620
       [a492] RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:4"
1621
       [a493] Effect="Deny">
1622
       [a494]
               <Description>
1623
                An Administrator shall not be permitted to read or write
       [a495]
1624
       [a496]
                 medical elements of a patient record in the
1625
       [a497]
                http://www.med.example.com/records.xsd namespace.
1626
       [a498] </Description>
1627
       [a499] <Target>
1628
       [a500]
               <Subjects>
1629
       [a501]
                 <Subject>
1630
       [a502]
                  <SubjectMatch
1631
       [a503]
                 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1632
       [a504]
                   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1633
       [a505]
                    administrator
1634
       [a506]
                    </AttributeValue>
1635
       [a507]
                    <SubjectAttributeDesignator AttributeId=</pre>
1636
       [a508]
                    "urn:oasis:names:tc:xacml:2.0:example:attribute:role"
1637
       [a509]
                    DataType="http://www.w3.org/2001/XMLSchema#string"/>
1638
       [a510]
                   </SubjectMatch>
1639
       [a511]
                 </Subject>
1640
       [a512]
                </Subjects>
1641
       [a513]
                <Resources>
1642
      [a514]
                 <Resource>
1643
      [a515]
                  <ResourceMatch
1644
       [a516]
                 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1645
      [a517]
                   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1646
       [a518]
                    urn:example:med:schemas:record
1647
       [a519]
                    </AttributeValue>
1648
       [a520]
                    <ResourceAttributeDesignator
1649
       AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
1650
       [a521]
                   DataType="http://www.w3.org/2001/XMLSchema#string"/>
1651
       [a522]
                   </ResourceMatch>
1652
       [a523]
                   <ResourceMatch
1653
       [a524]
                 MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
1654
       [a525]
                   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1655
       [a526]
                    /md:record/md:medical
1656
       [a527]
                   </AttributeValue>
1657
       [a528]
                    <ResourceAttributeDesignator</pre>
1658
       AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
1659
       [a529]
                   DataType="http://www.w3.org/2001/XMLSchema#string"/>
1660
       [a530]
                   </ResourceMatch>
1661
       [a531]
                  </Resource>
1662
       [a532]
                </Resources>
1663
       [a533]
                <Actions>
1664
       [a534]
                  <Action>
1665
       [a535]
                  <ActionMatch
1666
       [a536]
                  MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1667
       [a537]
                   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1668
       [a538]
1669
       [a539]
                    </AttributeValue>
1670
       [a540]
                   <ActionAttributeDesignator
1671
       AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1672
       [a541]
                   DataType="http://www.w3.org/2001/XMLSchema#string"/>
1673
       [a542]
                   </ActionMatch>
```

```
1674
      [a543]
                  </Action>
1675
       [a544]
                 <Action>
1676
       [a545]
                 <ActionMatch
1677
       [a546]
                 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1678
       [a547]
                   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1679
       [a548]
                    write
1680
       [a549]
                   </AttributeValue>
1681
       [a550]
                    <ActionAttributeDesignator
1682
       AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1683
       [a551]
                   DataType="http://www.w3.org/2001/XMLSchema#string"/>
1684
       [a552]
                   </ActionMatch>
1685
       [a553]
                  </Action>
1686
       [a554]
                </Actions>
1687
       [a555] </Target>
1688
       [a556] </Rule>
1689
       [a557] </Policy>
1690
       [a492] - [a493] The <Rule> element declaration.
1691
       [a493] Rule Effect. Every rule that evaluates to "True" emits the rule effect as its value. This
1692
       rule Effect is "Deny" meaning that according to this rule, access must be denied when it
1693
       evaluates to "True".
```

- 1694 [a494] [a498] Free form description of the *rule*.
- 1695 [a499] [a555] *Rule target*. The *Rule target* defines the set of *decision requests* that are applicable to the *rule*.
- 1697 [a502] [a510] The <SubjectMatch> element targets the $\it rule$ at $\it subjects$ whose
- 1698 "urn:oasis:names:tc:xacml:2.0:example:attribute:role" *subject attribute* is equal to
- 1699 "administrator".
- 1700 [a513] [a532] The <Resources> element contains one <Resource> element, which (in turn)
- 1701 contains two <ResourceMatch> elements. The target matches if the resource identified by the
- 1702 request *context* matches both *resource* match criteria.
- 1703 [a515]-[a522] The first <ResourceMatch> element targets the *rule* at *resources*1704 whose "urn:oasis:names:tc:xacml:2.0:resource:target-namespace" *resource attribute*
- is equal to "urn:example:med:schemas:record".
- 1706 [a523] [a530] The second <ResourceMatch> element targets the *rule* at XML elements that
- match the XPath expression "/md:record/md:medical".
- 1708 [a533] [a554] The <Actions> element contains two <Action> elements, each of which contains
- 1709 one <actionMatch> element. The *target* matches if the *action* identified in the request *context*
- 1710 matches either of the *action* match criteria.
- 1711 [a535] [a552] The <ActionMatch> elements target the *rule* at *actions* whose
- 1712 "urn:oasis:names:tc:xacml:1.0:action:action-id" action attribute is equal to "read" or "write".
- 1713 This *rule* does not have a <Condition> element.

1714 **4.2.4.5. Example PolicySet**

- 1715 This section uses the examples of the previous sections to illustrate the process of combining
- 1716 *policies*. The policy governing read access to medical elements of a record is formed from each of
- 1717 the four *rules* described in Section 4.2.3. In plain language, the combined rule is:
- 1718 Either the requestor is the patient; or
- the requestor is the parent or guardian and the patient is under 16; or

- the requestor is the primary care physician and a notification is sent to the patient; and
- the requestor is not an administrator.
- The following *policy set* illustrates the combined *policies*. *Policy* 3 is included by reference and *policy* 2 is explicitly included.

```
1724
       [a558] <?xml version="1.0" encoding="UTF-8"?>
1725
       [a559] <PolicySet
1726
       [a560] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
1727
       [a561] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1728
       xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
1729
       http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
1730
       [a562] PolicySetId=
1731
       [a563] "urn:oasis:names:tc:xacml:2.0:example:policysetid:1"
1732
       [a564] PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:
1733
       [a565] policy-combining-algorithm:deny-overrides">
1734
       [a566] <Description>
1735
       [a567]
               Example policy set.
1736
       [a568] </Description>
1737
       [a569] <Target>
1738
       [a570] <Resources>
1739
       [a571]
                <Resource>
1740
      [a572]
                  <ResourceMatch
       [a573]
1741
                MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1742
       [a574]
       [a575]
1743
                   urn:example:med:schema:records
       [a576]
1744
                   </AttributeValue>
1745
       [a577]
                   <ResourceAttributeDesignator AttributeId=</pre>
                  "urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
1746
       [a578]
1747
       [a579]
                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1748
      [a580]
                </ResourceMatch>
1749
      [a581]
                </Resource>
1750
      [a582] </Resources>
1751
      [a583] </Target>
1752
      [a584] <PolicyIdReference>
1753
      [a585] urn:oasis:names:tc:xacml:2.0:example:policyid:3
1754
      [a586] </PolicyIdReference>
1755
      [a587] <Policy
1756
       [a588] PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:2"
1757
       [a589] RuleCombiningAlgId=
1758
       [a590] "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
1759
       [a591]
              <Target/>
1760
       [a592]
               <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:1"</pre>
       [a593] Effect="Permit">
1761
1762
       [a594] </Rule>
1763
      [a595] <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:2"</pre>
1764
      [a596] Effect="Permit">
1765
       [a597] </Rule>
1766
       [a598] <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:4"
1767
       [a599] Effect="Deny">
1768
       [a600] </Rule>
1769
       [a601] </Policy>
1770
      [a602] </PolicySet>
1771
```

- 1772 [a559] [a565] The <PolicySet> element declaration. Standard XML namespace declarations are included.
- 1774 [a562] The PolicySetId attribute is used for identifying this *policy set* for possible inclusion in another *policy set*.

- 1776 [a564] The *policy combining algorithm* identifier. *Policies* and *policy sets* in this *policy set* are
- 1777 combined according to the specified *policy combining algorithm* when the *authorization*
- 1778 *decision* is computed.
- 1779 [a566] [a568] Free form description of the *policy set*.
- 1780 [a569] [a583] The *policy set* <Target> element defines the set of *decision requests* that are
- 1781 applicable to this <PolicySet > element.
- 1782 [a584] PolicyIdReference includes a *policy* by id.
- 1783 [a588] Policy 2 is explicitly included in this **policy set**. The **rules** in Policy 2 are omitted for
- 1784 clarity.

1786

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

5.1. Element <PolicySet>

- 1788 The <PolicySet> element is a top-level element in the XACML policy schema. <PolicySet> is
- an aggregation of other *policy sets* and *policies*. *Policy sets* MAY be included in an enclosing
- 1791 <PolicySetIdReference> element. Policies MAY be included in an enclosing <PolicySet>
- 1792 element either directly using the <Policy> element or indirectly using the
- 1794 A <PolicySet> element MAY be evaluated, in which case the evaluation procedure defined in
- 1795 Section 7.11 SHALL be used.
- 1796 If a <PolicySet> element contains references to other *policy sets* or *policies* in the form of
- 1797 URLs, then these references MAY be resolvable.
- 1798 **Policy sets** and **policies** included in a <PolicySet> element MUST be combined using the
- 1799 algorithm identified by the PolicyCombiningAlgId attribute. <PolicySet> is treated exactly
- 1800 like a <Policy> in all **policy combining algorithms**.
- The <Target> element defines the applicability of the <PolicySet> element to a set of *decision*
- 1802 requests. If the <Target> element within the <PolicySet> element matches the request
- 1803 context, then the <PolicySet> element MAY be used by the PDP in making its authorization
- 1804 *decision*. See Section 7.11.
- The <0bligations> element contains a set of *obligations* that MUST be fulfilled by the *PEP* in conjunction with the *authorization decision*. If the *PEP* does not understand, or cannot fulfill, any of the *obligations*, then it MUST act as if the *PDP* had returned a "Deny" *authorization decision*

```
1808 value. See Section 7.14.
```

```
1809
       <xs:element name="PolicySet" type="xacml:PolicySetType"/>
1810
       <xs:complexType name="PolicySetType">
1811
          <xs:sequence>
1812
            <xs:element ref="xacml:Description" minOccurs="0"/>
1813
             <xs:element ref="xacml:PolicySetDefaults" minOccurs="0"/>
             <xs:element ref="xacml:Target"/>
1814
1815
             <xs:choice minOccurs="0" maxOccurs="unbounded">
1816
               <xs:element ref="xacml:PolicySet"/>
```

```
1817
                 <xs:element ref="xacml:Policy"/>
1818
                 <xs:element ref="xacml:PolicySetIdReference"/>
1819
                 <xs:element ref="xacml:PolicyIdReference"/>
1820
                 <xs:element ref="xacml:CombinerParameters"/>
1821
                 <xs:element ref="xacml:PolicyCombinerParameters"/>
1822
                 <xs:element ref="xacml:PolicySetCombinerParameters"/>
1823
              </xs:choice>
1824
              <xs:element ref="xacml:Obligations" minOccurs="0"/>
1825
           </xs:sequence>
1826
           <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
1827
           <xs:attribute name="Version" type="xacml:VersionType" default="1.0"/>
           <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
1828
1829
        </xs:complexType>
1830
        The <PolicySet> element is of PolicySetType complex type.
1831
        The <PolicySet> element contains the following attributes and elements:
1832
        PolicySetId [Required]
               Policy set identifier. It is the responsibility of the PAP to ensure that no two policies
1833
               visible to the PDP have the same identifier. This MAY be achieved by following a
1834
1835
               predefined URN or URI scheme. If the policy set identifier is in the form of a URL, then it
1836
               MAY be resolvable.
1837
        Version [Default 1.0]
1838
               The version number of the PolicySet.
1839
        PolicyCombiningAlgId [Required]
1840
               The identifier of the policy-combining algorithm by which the <PolicySet>,
               <CombinerParameters>, <PolicyCombinerParameters> and
1841
1842
               <PolicySetCombinerParameters> components MUST be combined. Standard
               policy-combining algorithms are listed in Appendix C. Standard policy-combining
1843
1844
               algorithm identifiers are listed in Section B.10.
1845
        <Description> [Optional]
               A free-form description of the policy set.
1846
        <PolicySetDefaults>[Optional]
1847
1848
               A set of default values applicable to the policy set. The scope of the
1849
               <PolicySetDefaults> element SHALL be the enclosing policy set.
1850
        <Target> [Required]
1851
               The <Tarqet> element defines the applicability of a policy set to a set of decision
1852
1853
               The <Target> element MAY be declared by the creator of the <PolicySet> or it MAY be
               computed from the <Target> elements of the referenced <Policy> elements, either as
1854
1855
               an intersection or as a union.
1856
        <PolicySet> [Any Number]
1857
               A policy set that is included in this policy set.
1858
        <Policy> [Any Number]
               A policy that is included in this policy set.
1859
        access_control-xacml-2.0-core-spec-os
                                                                                     1 February 2005
        Copyright © OASIS Open 2004. All Rights Reserved.
                                                                                      Page 44 of 141
```

```
1860
        <PolicySetIdReference> [Any Number]
1861
               A reference to a policy set. that MUST be included in this policy set. If
               <PolicySetIdReference> is a URL, then it MAY be resolvable.
1862
1863
        <PolicyIdReference> [Any Number]
               A reference to a policy that MUST be included in this policy set. If the
1864
1865
               <PolicyIdReference> is a URL, then it MAY be resolvable.
1866
        <Obligations> [Optional]
1867
               Contains the set of <Obligation> elements. See Section 7.14 for a description of how
1868
               the set of obligations to be returned by the PDP shall be determined.
1869
        <CombinerParameters> [Optional]
1870
               Contains a sequence of <CombinerParameter> elements.
1871
        <PolicyCombinerParameters>[Optional]
1872
               Contains a sequence of <CombinerParameter> elements that are associated with a
               particular <Policy> or <PolicyIdReference> element within the <PolicySet>.
1873
1874
        <PolicySetCombinerParameters> [Optional]
1875
               Contains a sequence of <CombinerParameter> elements that are associated with a
1876
               particular <PolicySet> or <PolicySetIdReference> element within the
1877
               <PolicySet>.
           5.2.
                  Element < Description >
1878
1879
        The <Description> element contains a free-form description of the <PolicySet>, <Policy>
1880
        or <Rule> element. The <Description> element is of xs:string simple type.
1881
          <xs:element name="Description" type="xs:string"/>
                  Element < Policy Set Defaults >
1882
        The <PolicySetDefaults> element SHALL specify default values that apply to the
1883
1884
        <PolicySet> element.
1885
        <xs:element name="PolicySetDefaults" type="xacml:DefaultsType"/>
1886
        <xs:complexType name="DefaultsType">
1887
           <xs:sequence>
1888
             <xs:choice>
1889
                <xs:element ref="xacml:XPathVersion" minOccurs="0"/>
1890
             </xs:choice>
1891
           </xs:sequence>
1892
        </xs:complexType>
1893
        <PolicySetDefaults> element is of DefaultsType complex type.
1894
        The <PolicySetDefaults> element contains the following elements:
1895
        <XPathVersion> [Optional]
1896
               Default XPath version.
```

5.4. Element <XPathVersion>

The <XPathVersion> element SHALL specify the version of the XPath specification to be used by

<AttributeSelector> elements and XPath-based functions in the policy set or policy.


```
1900 <xs:element name="XPathVersion" type="xs:anyURI"/>
```

- The URI for the XPath 1.0 specification is "http://www.w3.org/TR/1999/Rec-xpath-19991116". The
- 1902 <XPathVersion> element is REQUIRED if the XACML enclosing *policy set* or *policy* contains
- 1903 AttributeSelector> elements or XPath-based functions.

5.5. Element <Target>

- 1905 The <Target> element identifies the set of *decision requests* that the parent element is intended
- 1906 to evaluate. The <Target> element SHALL appear as a child of a <PolicySet> and <Policy>
- 1907 element and MAY appear as a child of a <Rule> element. It contains definitions for **subjects**,
- 1908 resources, actions and environments.
- 1909 The <Target> element SHALL contain a conjunctive sequence of <Subjects>, <Resources>
- 1910 <Actions> and <Environments> elements. For the parent of the <Target> element to be
- 1911 applicable to the *decision request*, there MUST be at least one positive match between each
- 1912 section of the <Target> element and the corresponding section of the <xacml-
- 1913 context: Request > element.

1897

1904

```
1914
       <xs:element name="Target" type="xacml:TargetType"/>
1915
       <xs:complexType name="TargetType">
1916
          <xs:sequence>
1917
             <xs:element ref="xacml:Subjects" minOccurs="0"/>
1918
             <xs:element ref="xacml:Resources" minOccurs="0"/>
1919
             <xs:element ref="xacml:Actions" minOccurs="0"/>
1920
             <xs:element ref="xacml:Environments" minOccurs="0"/>
1921
          </xs:sequence>
1922
       </xs:complexType>
```

- 1923 The <Target> element is of **TargetType** complex type.
- 1924 The <Target> element contains the following elements:
- 1925 <Subjects> [Optional]
- Matching specification for the *subject attributes* in the *context*. If this element is missing, then the *target* SHALL match all *subjects*.
- 1928 <Resources> [Optional]
- Matching specification for the *resource attributes* in the *context*. If this element is missing, then the *target* SHALL match all *resources*.
- 1931 <Actions>[Optional]
- Matching specification for the *action attributes* in the *context*. If this element is missing, then the *target* SHALL match all *actions*.
- 1934 <Environments> [Optional]
- 1935 Matching specification for the *environment attributes* in the *context*. If this element is missing, then the *target* SHALL match all *environments*.

5.6. Element <Subjects>

1938 The <Subjects> element SHALL contain a *disjunctive sequence* of <Subject> elements.

- 1945 The <Subjects> element is of **SubjectsType** complex type.
- 1946 The <Subjects> element contains the following elements:
- 1947 <Subject> [One to Many, Required]
- 1948 See Section 5.7.

1937

1949

1961

1962

1963

1964

1965

1966

5.7. Element <Subject>

1950 The <Subject> element SHALL contain a *conjunctive sequence* of <SubjectMatch> 1951 elements.

- 1958 The <Subject> element is of **SubjectType** complex type.
- 1959 The <Subject> element contains the following elements:
- 1960 <SubjectMatch> [One to Many]

A **conjunctive sequence** of individual matches of the **subject attributes** in the request **context** and the embedded **attribute** values. See Section 5.8.

5.8. Element <SubjectMatch>

The <SubjectMatch> element SHALL identify a set of **subject**-related entities by matching **attribute** values in a <xacml-context: Subject> element of the request **context** with the embedded **attribute** value.

```
1967
       <xs:element name="SubjectMatch" type="xacml:SubjectMatchType"/>
1968
       <xs:complexType name="SubjectMatchType">
1969
          <xs:sequence>
1970
             <xs:element ref="xacml:AttributeValue"/>
1971
             <xs:choice>
1972
               <xs:element ref="xacml:SubjectAttributeDesignator"/>
1973
               <xs:element ref="xacml:AttributeSelector"/>
1974
             </xs:choice>
1975
          </xs:sequence>
1976
          <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
1977
       </xs:complexType>
```

- 1978 The <SubjectMatch> element is of **SubjectMatchType** complex type.
- 1979 The <SubjectMatch> element contains the following attributes and elements:

```
1980
        MatchId [Required]
1981
               Specifies a matching function. The value of this attribute MUST be of type xs:anyURI with
1982
               legal values documented in Section 7.5.
1983
        <xacml:AttributeValue>[Required]
1984
               Embedded attribute value.
1985
        <SubjectAttributeDesignator> [Required choice]
               MAY be used to identify one or more attribute values in a <Subject> element of the
1986
1987
               request context.
1988
        <a href="#"><AttributeSelector> [Required choice]</a>
1989
               MAY be used to identify one or more attribute values in the request context. The XPath
1990
               expression SHOULD resolve to an attribute in a <Subject> element of the request
1991
               context.
1992
           5.9.
                  Element < Resources >
1993
        The <Resources> element SHALL contain a disjunctive sequence of <Resource> elements.
1994
        <xs:element name="Resources" type="xacml:ResourcesType"/>
1995
        <xs:complexType name="ResourcesType">
1996
           <xs:sequence>
1997
              <xs:element ref="xacml:Resource" maxOccurs="unbounded"/>
1998
           </xs:sequence>
1999
        </xs:complexType>
2000
        The <Resources> element is of ResourcesType complex type.
2001
        The <Resources> element contains the following elements:
2002
        <Resource> [One to Many, Required]
2003
               See Section 5.10.
           5.10. Element < Resource >
2004
2005
        The <Resource> element SHALL contain a conjunctive sequence of <ResourceMatch>
2006
        elements.
2007
        <xs:element name="Resource" type="xacml:ResourceType"/>
2008
        <xs:complexType name="ResourceType">
2009
           <xs:sequence>
2010
             <xs:element ref="xacml:ResourceMatch" maxOccurs="unbounded"/>
2011
           </xs:sequence>
2012
        </xs:complexType>
2013
        The <Resource> element is of ResourceType complex type.
2014
        The <Resource> element contains the following elements:
2015
        <ResourceMatch> [One to Many]
2016
               A conjunctive sequence of individual matches of the resource attributes in the request
2017
               context and the embedded attribute values. See Section 5.11.
```

5.11. Element < Resource Match >

The <ResourceMatch> element SHALL identify a set of *resource*-related entities by matching attribute values in the <xacml-context:Resource> element of the request *context* with the embedded attribute value.

```
2022
       <xs:element name="ResourceMatch" type="xacml:ResourceMatchType"/>
2023
       <xs:complexType name="ResourceMatchType">
2024
          <xs:sequence>
2025
             <xs:element ref="xacml:AttributeValue"/>
2026
             <xs:choice>
2027
                <xs:element ref="xacml:ResourceAttributeDesignator"/>
2028
                <xs:element ref="xacml:AttributeSelector"/>
2029
             </xs:choice>
2030
          </xs:sequence>
2031
          <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
2032
       </xs:complexType>
```

- 2033 The <ResourceMatch> element is of ResourceMatchType complex type.
- 2034 The <ResourceMatch> element contains the following attributes and elements:
- 2035 MatchId [Required]

2018

- Specifies a matching function. Values of this attribute MUST be of type **xs:anyURI**, with legal values documented in Section 7.5.
- 2038 <xacml:AttributeValue> [Required]
- 2039 Embedded attribute value.
- 2040 <ResourceAttributeDesignator> [Required Choice]
- 2041 MAY be used to identify one or more *attribute* values in the <Resource> element of the request *context*.
- 2043 AttributeSelector [Required Choice]
- MAY be used to identify one or more *attribute* values in the request *context*. The XPath expression SHOULD resolve to an *attribute* in the <Resource> element of the request *context*.

2047 5.12. Element < Actions >

2048 The <actions> element SHALL contain a disjunctive sequence of <action> elements.

- 2055 The <Actions> element is of **ActionsType** complex type.
- 2056 The <Actions> element contains the following elements:
- 2057 <Action> [One to Many, Required]
- 2058 See Section 5.13.

5.13. Element <Action>

2060 The <action> element SHALL contain a conjunctive sequence of <actionMatch> elements.

- 2067 The <action> element is of ActionType complex type.
- 2068 The <action> element contains the following elements:
- 2069 <ActionMatch> [One to Many]

2059

2072

2070 A *conjunctive sequence* of individual matches of the *action attributes* in the request 2071 *context* and the embedded *attribute* values. See Section 5.14.

5.14. Element < Action Match >

The <actionMatch> element SHALL identify a set of *action*-related entities by matching *attribute* values in the <acml-context:Action> element of the request *context* with the embedded *attribute* value.

```
2076
       <xs:element name="ActionMatch" type="xacml:ActionMatchType"/>
2077
       <xs:complexType name="ActionMatchType">
2078
          <xs:sequence>
2079
             <xs:element ref="xacml:AttributeValue"/>
2080
             <xs:choice>
2081
                <xs:element ref="xacml:ActionAttributeDesignator"/>
2082
                <xs:element ref="xacml:AttributeSelector"/>
2083
             </xs:choice>
2084
          </xs:sequence>
2085
          <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
2086
       </xs:complexType>
```

- 2087 The <ActionMatch> element is of **ActionMatchType** complex type.
- 2088 The <actionMatch> element contains the following attributes and elements:
- 2089 MatchId [Required]
- Specifies a matching function. The value of this attribute MUST be of type **xs:anyURI**, with legal values documented in Section 7.5.
- 2092 <xacml:AttributeValue> [Required]
- 2093 Embedded attribute value.
- 2094 <actionAttributeDesignator> [Required Choice]
- 2095 MAY be used to identify one or more *attribute* values in the <Action> element of the request *context*.
- 2097 AttributeSelector> [Required Choice]
- 2098 MAY be used to identify one or more *attribute* values in the request *context*. The XPath expression SHOULD resolve to an *attribute* in the <action> element of the *context*.

5.15. Element < Environments >

- The <Environments> element SHALL contain a *disjunctive sequence* of <Environment> elements.
- 2109 The <Environments> element is of EnvironmentsType complex type.
- 2110 The <Environments> element contains the following elements:
- 2111 <Environment> [One to Many, Required]
- 2112 See Section 5.16.

2100

2113

2127

5.16. Element < Environment>

- 2114 The <Environment> element SHALL contain a conjunctive sequence of
- 2115 <EnvironmentMatch> elements.

- 2122 The <Environment> element is of **EnvironmentType** complex type.
- 2123 The <Environment> element contains the following elements:
- 2124 <EnvironmentMatch> [One to Many]
- A *conjunctive sequence* of individual matches of the *environment* attributes in the request *context* and the embedded *attribute* values.

5.17. Element < Environment Match >

The <EnvironmentMatch> element SHALL identify an environment by matching attribute values in the <xacml-context:Environment> element of the request context with the embedded attribute value.

```
2131
       <xs:element name="EnvironmentMatch" type="xacml:EnvironmentMatchType"/>
2132
       <xs:complexType name="EnvironmentMatchType">
2133
          <xs:sequence>
2134
             <xs:element ref="xacml:AttributeValue"/>
2135
2136
               <xs:element ref="xacml:EnvironmentAttributeDesignator"/>
2137
               <xs:element ref="xacml:AttributeSelector"/>
2138
             </xs:choice>
2139
          </xs:sequence>
2140
          <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
2141
       </xs:complexType>
```

2142 The <EnvironmentMatch> element is of EnvironmentMatchType complex type.

2143	The <environmentmatch> element contains the following attributes and elements:</environmentmatch>
2144	MatchId [Required]
2145 2146	Specifies a matching function. The value of this attribute MUST be of type xs:anyURI , with legal values documented in Section A.3.
2147	<pre><xacml:attributevalue> [Required]</xacml:attributevalue></pre>
2148	Embedded attribute value.
2149	<pre><environmentattributedesignator> [Required Choice]</environmentattributedesignator></pre>
2150 2151	MAY be used to identify one or more <i>attribute</i> values in the <environment> element of the request <i>context</i>.</environment>
2152	<attributeselector< a="">> [Required Choice]</attributeselector<>
2153 2154 2155	MAY be used to identify one or more attribute values in the request context . The XPath expression SHOULD resolve to an attribute in the <environment> element of the request context.</environment>
2156	5.18. Element <policysetidreference></policysetidreference>
2157 2158 2159 2160	The <pre><pre>PolicySetIdReference> element SHALL be used to reference a <policyset> element by id. If <pre><pre><pre>FolicySetIdReference> is a URL</pre>, then it MAY be resolvable to the <pre><pre><pre><pre>FolicySet></pre> element. However, the mechanism for resolving a policy set reference to the corresponding policy set is outside the scope of this specification.</pre></pre></pre></pre></pre></policyset></pre></pre>
2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173	<pre><xs:element name="PolicySetIdReference" type="xacml:IdReferenceType"></xs:element> xs:complexType name="IdReferenceType"></pre>
2174	Element <policysetidreference> is of xacml:IdReferenceType complex type.</policysetidreference>
2175	IdReferenceType extends the xs:anyURI type with the following attributes:
2176	Version [Optional]
2177	Specifies a matching expression for the version of the <i>policy set</i> referenced.
2178	EarliestVersion [Optional]
2179 2180	Specifies a matching expression for the earliest acceptable version of the <i>policy set</i> referenced.
2181	LatestVersion [Optional]
2182 2183	Specifies a matching expression for the latest acceptable version of the <i>policy set</i> referenced.

- 2184 The matching operation is defined in Section 5.21. Any combination of these attributes MAY be
- 2185 present in a <PolicySetIdReference>. The referenced policy set MUST match all
- 2186 expressions. If none of these attributes is present, then any version of the policy set is acceptable.
- 2187 In the case that more than one matching version can be obtained, then the most recent one
- 2188 SHOULD be used.

2196

2205

2216

2218

5.19. Element <PolicyIdReference>

- 2190
- 2191 by id. If <PolicyIdReference> is a URL, then it MAY be resolvable to the <Policy> element.
- 2192 However, the mechanism for resolving a *policy* reference to the corresponding *policy* is outside
- 2193 the scope of this specification.
- 2194 <xs:element name="PolicyIdReference" type="xacml:IdReferenceType"/>
- 2195 Element <PolicyIdReference> is of xacml:IdReferenceType complex type (see Section 5.18).

5.20. Simple type VersionType

2197 Elements of this type SHALL contain the version number of the **policy** or **policy set**.

```
2198
       <xs:simpleType name="VersionType">
2199
          <xs:restriction base="xs:string">
2200
             <xs:pattern value="(\d+\.)*\d+"/>
2201
          </xs:restriction>
2202
       </xs:simpleType>
```

2203 The version number is expressed as a sequence of decimal numbers, each separated by a period 2204 (.). 'd+' represents a sequence of one or more decimal digits.

5.21. Simple type VersionMatchType

2206 Elements of this type SHALL contain a restricted regular expression matching a version number 2207 (see Section 5.20). The expression SHALL match versions of a referenced policy or policy set 2208 that are acceptable for inclusion in the referencing *policy* or *policy set*.

```
2209
       <xs:simpleType name="VersionMatchType">
2210
          <xs:restriction base="xs:string">
2211
             <xs:pattern value="((\d+|\*)\.)*(\d+|\*|\+)"/>
2212
          </xs:restriction>
2213
       </xs:simpleType>
```

A version match is '.'-separated, like a version string. A number represents a direct numeric match. 2214

2215 A '*' means that any single number is valid. A '+' means that any number, and any subsequent

numbers, are valid. In this manner, the following four patterns would all match the version string

2217 '1.2.3': '1.2.3', '1.*.3', '1.2.*' and '1.+'.

5.22. Element <Policy>

- 2219 The <Policy> element is the smallest entity that SHALL be presented to the **PDP** for evaluation.
- 2220 A <Policy> element MAY be evaluated, in which case the evaluation procedure defined in
- Section 7.10 SHALL be used. 2221
- 2222 The main components of this element are the <Target>, <Rule>, <CombinerParameters>,
- 2223 <RuleCombinerParameters> and <Obligations> elements and the RuleCombiningAlgId
- 2224 attribute.

- 2225 The <Target> element defines the applicability of the <Policy> element to a set of *decision*
- 2226 requests. If the <Target> element within the <Policy> element matches the request context,
- 2227 then the <Policy> element MAY be used by the **PDP** in making its **authorization decision**. See
- 2228 Section 7.10.
- 2229 The <Policy> element includes a sequence of choices between <VariableDefinition> and
- 2230 <Rule> elements.
- 2231 **Rules** included in the <Policy> element MUST be combined by the algorithm specified by the
- 2232 RuleCombiningAlgId attribute.
- The <Obligations > element contains a set of *obligations* that MUST be fulfilled by the *PEP* in
- 2234 conjunction with the *authorization decision*.

```
2235
       <xs:element name="Policy" type="xacml:PolicyType"/>
2236
       <xs:complexType name="PolicyType">
2237
          <xs:sequence>
2238
            <xs:element ref="xacml:Description" minOccurs="0"/>
2239
            <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>
2240
            <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>
2241
            <xs:element ref="xacml:Target"/>
2242
            <xs:choice maxOccurs="unbounded">
2243
               <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>
2244
               <xs:element ref="xacml:RuleCombinerParameters" minOccurs="0"/>
2245
               <xs:element ref="xacml:VariableDefinition"/>
2246
               <xs:element ref="xacml:Rule"/>
2247
            </xs:choice>
2248
            <xs:element ref="xacml:Obligations" minOccurs="0"/>
2249
          </xs:sequence>
2250
          <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
2251
          <xs:attribute name="Version" type="xacml:VersionType" default="1.0"/>
2252
          <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
2253
       </xs:complexType>
```

- 2254 The <Policy> element is of **PolicyType** complex type.
- 2255 The <Policy> element contains the following attributes and elements:
- 2256 PolicyId [Required]
- Policy identifier. It is the responsibility of the PAP to ensure that no two policies visible to
 the PDP have the same identifier. This MAY be achieved by following a predefined URN or
 URI scheme. If the policy identifier is in the form of a URL, then it MAY be resolvable.
- 2260 Version [Default 1.0]
- The version number of the *Policy*.
- 2262 RuleCombiningAlgId [Required]
- 2263 The identifier of the *rule-combining algorithm* by which the <Policy>,
- 2264 <CombinerParameters> and <RuleCombinerParameters> components MUST be
 2265 combined. Standard *rule-combining algorithms* are listed in Appendix C. Standard *rule-combining algorithm* identifiers are listed in Section B.10.

<Description> [Optional]

- 2268 A free-form description of the *policy*. See Section 5.2.
- 2269 <PolicyDefaults>[Optional]

2267

```
2270
               Defines a set of default values applicable to the policy. The scope of the
2271
               <PolicyDefaults> element SHALL be the enclosing policy.
2272
        <CombinerParameters> [Optional]
2273
               A sequence of parameters to be used by the rule-combining algorithm.
2274
        <RuleCombinerParameters> [Optional]
2275
               A sequence of parameters to be used by the rule-combining algorithm.
2276
        <Target> [Required]
2277
               The <Target> element defines the applicability of a <Policy> to a set of decision requests.
2278
               The <Target> element MAY be declared by the creator of the <Policy> element, or it
               MAY be computed from the <Target> elements of the referenced <Rule> elements either
2279
               as an intersection or as a union.
2280
2281
        <VariableDefinition> [Any Number]
2282
               Common function definitions that can be referenced from anywhere in a rule where an
2283
               expression can be found.
2284
        <Rule> [Any Number]
2285
               A sequence of rules that MUST be combined according to the RuleCombiningAlgId
2286
               attribute. Rules whose <Target> elements match the decision request MUST be
2287
               considered. Rules whose <Target> elements do not match the decision request SHALL
2288
               be ignored.
2289
        <Obligations> [Optional]
2290
               A conjunctive sequence of obligations that MUST be fulfilled by the PEP in conjunction
2291
               with the authorization decision. See Section 7.14 for a description of how the set of
               obligations to be returned by the PDP SHALL be determined.
2292
           5.23. Element <PolicyDefaults>
2293
2294
        The <PolicyDefaults> element SHALL specify default values that apply to the <Policy>
2295
        element.
2296
        <xs:element name="PolicyDefaults" type="xacml:DefaultsType"/>
2297
        <xs:complexType name="DefaultsType">
2298
           <xs:sequence>
2299
              <xs:choice>
2300
                 <xs:element ref="xacml:XPathVersion" minOccurs="0"/>
2301
              </xs:choice>
2302
           </xs:sequence>
2303
        </xs:complexType>
        <PolicyDefaults> element is of DefaultsType complex type.
2304
2305
        The <PolicyDefaults> element contains the following elements:
2306
        <XPathVersion> [Optional]
2307
               Default XPath version.
```

5.24. Element < Combiner Parameters>

2309 The <CombinerParameters> element conveys parameters for a policy- or rule-combining

2310 *algorithm*.

2330

- 2311 If multiple <CombinerParameters> elements occur within the same policy or policy set, they
- 2312 SHALL be considered equal to one <CombinerParameters> element containing the
- 2313 concatenation of all the sequences of <CombinerParameters> contained in all the aforementioned
- 2314 <CombinerParameters> elements, such that the order of occurence of the
- 2315 <CominberParameters> elements is preserved in the concatenation of the
- 2316 <CombinerParameter> elements.
- 2317 Note that none of the *combining algorithms* specified in XACML 2.0 is parameterized.

- 2325 The <CombinerParameters> element is of CombinerParametersType complex type.
- 2326 The <CombinerParameters> element contains the following elements:
- 2327 <CombinerParameter> [Any Number]
- 2328 A single parameter. See Section 5.25.
- 2329 Support for the <CombinerParameters> element is optional.

5.25. Element < Combiner Parameter >

The <CombinerParameter> element conveys a single parameter for a *policy*- or *rule-combining algorithm*.

- 2340 The <CombinerParameter> element is of CombinerParameterType complex type.
- 2341 The <CombinerParameter> element contains the following attribute:
- 2342 ParameterName [Required]
- 2343 The identifier of the parameter.
- 2344 AttributeValue [Required]
- The value of the parameter.
- 2346 Support for the <CombinerParameter> element is optional.

5.26. Element < Rule Combiner Parameters >

- 2348 The <RuleCombinerParameters> element conveys parameters associated with a particular
- 2349 *rule* within a *policy* for a *rule-combining algorithm*.
- 2350 Each <RuleCombinerParameters > element MUST be associated with a *rule* contained within
- 2351 the same *policy*. If multiple <RuleCombinerParameters> elements reference the same *rule*,
- 2352 they SHALL be considered equal to one <RuleCombinerParameters> element containing the
- 2353 concatenation of all the sequences of <CombinerParameters> contained in all the aforementioned
- 2354 <RuleCombinerParameters> elements, such that the order of occurence of the
- 2355 <RuleCominberParameters> elements is preserved in the concatenation of the
- 2356 <CombinerParameter> elements.

2347

2367

2372

Note that none of the *rule-combining algorithms* specified in XACML 2.0 is parameterized.

```
2358
       <xs:element name="RuleCombinerParameters"</pre>
2359
       type="xacml:RuleCombinerParametersType"/>
2360
       <xs:complexType name="RuleCombinerParametersType">
2361
          <xs:complexContent>
2362
             <xs:extension base="xacml:CombinerParametersType">
2363
                <xs:attribute name="RuleIdRef" type="xs:string" use="required"/>
2364
2365
          </xs:complexContent>
2366
       </xs:complexType>
```

- The <RuleCombinerParameters> element contains the following elements:
- 2368 RuleIdRef [Required]
- 2369 The identifier of the <Rule> contained in the *policy*.
- Support for the <RuleCombinerParameters > element is optional, only if support for *combiner* parameters is optional.

5.27. Element < Policy Combiner Parameters >

- The <PolicyCombinerParameters> element conveys *parameters* associated with a particular policy within a policy set for a policy-combining algorithm.
- 2375 Each <PolicyCombinerParameters> element MUST be associated with a *policy* contained
- 2376 within the same *policy set*. If multiple <PolicyCombinerParameters> elements reference the
- 2377 same *policy*, they SHALL be considered equal to one <PolicyCombinerParameters> element
- 2378 containing the concatenation of all the sequences of <CombinerParameters> contained in all the
- 2379 aforementioned <PolicyCombinerParameters> elements, such that the order of occurence of
- 2380 the <PolicyCominberParameters> elements is preserved in the concatenation of the
- 2381 <CombinerParameter> elements.
- Note that none of the *policy-combining algorithms* specified in XACML 2.0 is parameterized.

```
2383
       <xs:element name="PolicyCombinerParameters"</pre>
2384
       type="xacml:PolicyCombinerParametersType"/>
2385
       <xs:complexType name="PolicyCombinerParametersType">
2386
          <xs:complexContent>
2387
             <xs:extension base="xacml:CombinerParametersType">
2388
                <xs:attribute name="PolicyIdRef" type="xs:anyURI" use="required"/>
2389
             </xs:extension>
2390
          </xs:complexContent>
2391
       </xs:complexType>
```

2392 2393	The <policycombinerparameters> element is of PolicyCombinerParametersType complex type.</policycombinerparameters>
2394	The <policycombinerparameters> element contains the following elements:</policycombinerparameters>
2395	PolicyIdRef [Required]
2396 2397	The identifier of a <policy> or the value of a <policyidreference> contained in the policy set.</policyidreference></policy>
2398 2399	Support for the <policycombinerparameters> element is optional, only if support for combiner parameters is optional.</policycombinerparameters>
2400	5.28. Element <policysetcombinerparameters></policysetcombinerparameters>
2401 2402	The <policysetcombinerparameters> element conveys <i>parameters</i> associated with a particular <i>policy set</i> within a <i>policy set</i> for a <i>policy-combining algorithm</i>.</policysetcombinerparameters>
2403 2404 2405 2406 2407 2408 2409 2410	Each <policysetcombinerparameters> element MUST be associated with a <i>policy set</i> contained within the same <i>policy set</i>. If multiple <policysetcombinerparameters> elements reference the same <i>policy set</i>, they SHALL be considered equal to one <policysetcombinerparameters> element containing the concatenation of all the sequences of <combinerparameters> contained in all the aforementioned <policysetcombinerparameters> elements, such that the order of occurence of the <policysetcominberparameters> elements is preserved in the concatenation of the <combinerparameter> elements.</combinerparameter></policysetcominberparameters></policysetcombinerparameters></combinerparameters></policysetcombinerparameters></policysetcombinerparameters></policysetcombinerparameters>
2411	Note that none of the <i>policy-combining algorithms</i> specified in XACML 2.0 is parameterized.
2412 2413 2414 2415 2416 2417 2418 2419 2420	<pre><xs:element name="PolicySetCombinerParameters" type="xacml:PolicySetCombinerParametersType"></xs:element> <xs:complextype name="PolicySetCombinerParametersType"></xs:complextype></pre>
2421	The <pre><pre><pre><pre>The <pre><pre><pre>PolicySetCombinerParameters</pre><pre><pre><pre>pe</pre></pre></pre></pre></pre></pre></pre></pre></pre>
2422	complex type.
2423	The <policysetcombinerparameters> element contains the following elements:</policysetcombinerparameters>
2424	PolicySetIdRef [Required]
2425 2426	The identifier of a <policyset> or the value of a <policysetidreference> contained in the <i>policy set</i>.</policysetidreference></policyset>
2427 2428	Support for the <policysetcombinerparameters> element is optional, only if support for combiner parameters is optional.</policysetcombinerparameters>
2429	5.29. Element <rule></rule>
2430	The <pulle> element SHALL define the individual rules in the notice. The main components of</pulle>

this element are the <Target> and <Condition> elements and the Effect attribute.

2431

```
2432
        A <Rule> element MAY be evaluated, in which case the evaluation procedure defined in Section
        7.9 SHALL be used.
2433
```

```
2434
       <xs:element name="Rule" type="xacml:RuleType"/>
2435
       <xs:complexType name="RuleType">
2436
          <xs:sequence>
2437
             <xs:element ref="xacml:Description" minOccurs="0"/>
2438
             <xs:element ref="xacml:Target" minOccurs="0"/>
2439
             <xs:element ref="xacml:Condition" minOccurs="0"/>
2440
2441
          <xs:attribute name="RuleId" type="xs:string" use="required"/>
2442
          <xs:attribute name="Effect" type="xacml:EffectType" use="required"/>
2443
       </xs:complexType>
```

- 2444 The <Rule> element is of RuleType complex type.
- 2445 The <Rule> element contains the following attributes and elements:
- 2446 RuleId [Required]
- 2447 A string identifying this *rule*.
- 2448 Effect [Required]
- 2449 **Rule effect**. The value of this attribute is either "Permit" or "Deny".
- 2450 <Description> [Optional]
- 2451 A free-form description of the *rule*.
- 2452 <Target> [Optional]
- 2453 Identifies the set of decision requests that the <Rule> element is intended to evaluate. If 2454 this element is omitted, then the target for the <Rule> SHALL be defined by the 2455
- <Target> element of the enclosing <Policy> element. See Section 7.6 for details.
- 2456 <Condition> [Optional]

2467

2457 A *predicate* that MUST be satisfied for the *rule* to be assigned its Effect value.

5.30. Simple type EffectType

2459 The EffectType simple type defines the values allowed for the Effect attribute of the <Rule> 2460 element and for the Fulfillon attribute of the <Obligation> element.

```
2461
       <xs:simpleType name="EffectType">
2462
          <xs:restriction base="xs:string">
2463
             <xs:enumeration value="Permit"/>
2464
             <xs:enumeration value="Deny"/>
2465
          </xs:restriction>
2466
       </xs:simpleType>
```

5.31. Element < Variable Definition >

The <VariableDefinition> element SHALL be used to define a value that can be referenced 2468 2469 by a <VariableReference> element. The name supplied for its VariableId attribute SHALL 2470 NOT occur in the VariableId attribute of any other <VariableDefinition> element within the 2471 encompassing policy. The <VariableDefinition> element MAY contain undefined 2472 <VariableReference> element, but if it does, a corresponding <VariableDefinition> element 2473 MUST be defined later in the encompassing policy. <VariableDefinition> elements MAY be

grouped together or MAY be placed close to the reference in the encompassing *policy*. There
MAY be zero or more references to each <VariableDefinition> element.

- The <VariableDefinition> element is of VariableDefinitionType complex type. The <VariableDefinition> element has the following elements and attributes:
- 2485 <Expression> [Required]
- 2486 Any element of **ExpressionType** complex type.
- 2487 VariableId [Required]

2489

2490

2491

2492

2493

2494

2495

2496

2506

2507

2508

2509

2488 The name of the variable definition.

5.32. Element < Variable Reference >

The <VariableReference> element is used to reference a value defined within the same encompassing <Policy> element. The <VariableReference> element SHALL refer to the <VariableDefinition> element by string equality on the value of their respective VariableId attributes. There SHALL exist one and only one <VariableDefinition> within the same encompassing <Policy> element to which the <VariableReference> refers. There MAY be zero or more <VariableReference> elements that refer to the same <VariableDefinition> element.

```
2497
       <xs:element name="VariableReference" type="xacml:VariableReferenceType"</pre>
2498
       substitutionGroup="xacml:Expression"/>
2499
       <xs:complexType name="VariableReferenceType">
2500
          <xs:complexContent>
2501
             <xs:extension base="xacml:ExpressionType">
2502
               <xs:attribute name="VariableId" type="xs:string" use="required"/>
2503
             </xs:extension>
2504
          </xs:complexContent>
2505
       </xs:complexType>
```

- The <VariableReference> element is of the VariableReferenceType complex type, which is of the ExpressionType complex type and is a member of the <Expression> element substitution group. The <VariableReference> element MAY appear any place where an <Expression> element occurs in the schema.
- 2510 The <VariableReference> element has the following attributes:
- 2511 VariableId [Required]
- 2512 The name used to refer to the value defined in a <VariableDefinition> element.

2513 5.33. Element < Expression >

- The <Expression> element is not used directly in a *policy*. The <Expression> element signifies that an element that extends the **ExpressionType** and is a member of the
- 2516 <Expression> element substitution group SHALL appear in its place.
- 2517 2517 content cased and a group of wall appear in the place.
 2517 content name="Expression" type="xacml:ExpressionType" abstract="true"/>

```
2518 <xs:complexType name="ExpressionType" abstract="true"/>
```

2519 The following elements are in the <Expression> element substitution group:

```
2520 <Apply>, <AttributeSelector>, <AttributeValue>, <Function>,
```

- 2521 <VariableReference>, <ActionAttributeDesignator>,
- 2522 <ResourceAttributeDesignator>, <SubjectAttributeDesignator> and
- 2523 <EnvironmentAttributeDesignator>.

2524

2536

2537

2538

2539

5.34. Element < Condition>

The <Condition> element is a Boolean function over *subject*, *resource*, *action* and *environment attributes* or functions of *attributes*.

- 2533 The <Condition> contains one <Expression> element, with the restriction that the
- 2534 <Expression> return data-type MUST be "http://www.w3.org/2001/XMLSchema#boolean".
- 2535 Evaluation of the <Condition> element is described in Section 7.8.

5.35. Element < Apply>

The <Apply> element denotes application of a function to its arguments, thus encoding a function call. The <Apply> element can be applied to any combination of the members of the <Expression> element substitution group. See Section 5.33.

```
2540
        <xs:element name="Apply" type="xacml:ApplyType"</pre>
2541
       substitutionGroup="xacml:Expression"/>
2542
       <xs:complexType name="ApplyType">
2543
          <xs:complexContent>
2544
             <xs:extension base="xacml:ExpressionType">
2545
                <xs:sequence>
2546
                  <xs:element ref="xacml:Expression" minOccurs="0"</pre>
2547
       maxOccurs="unbounded"/>
2548
               </xs:sequence>
2549
                <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
2550
             </xs:extension>
2551
          </xs:complexContent>
2552
       </xs:complexType>
```

- 2553 The <Apply> element is of **ApplyType** complex type.
- 2554 The <Apply> element contains the following attributes and elements:
- 2555 FunctionId [Required]
- The identifier of the function to be applied to the arguments. XACML-defined functions are described in Appendix A.
- 2558 <Expression> [Optional]
- 2559 Arguments to the function, which may include other functions.

5.36. Element <Function>

The <Function> element SHALL be used to name a function as an argument to the function defined by the parent <Apply> element. In the case where the parent <Apply> element is a higher-order *bag* function, the named function is applied to every element of the *bag* or *bags* identified in the other arguments of the parent element. The higher-order *bag* functions are described in Section A3A.3.12.

```
2566
       <xs:element name="Function" type="xacml:FunctionType"</pre>
2567
       substitutionGroup="xacml:Expression"/>
2568
       <xs:complexType name="FunctionType">
2569
          <xs:complexContent>
2570
             <xs:extension base="xacml:ExpressionType">
2571
                <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
2572
             </xs:extension>
2573
          </xs:complexContent>
2574
       </xs:complexType>
```

- 2575 The Function element is of **FunctionType** complex type.
- 2576 The Function element contains the following attributes:
- 2577 FunctionId [Required]

2560

2561

2562

2563 2564

2565

2579

2598

2599

2600

2601

2578 The identifier of the function.

5.37. Complex type AttributeDesignatorType

- The **AttributeDesignatorType** complex type is the type for elements that identify **attributes** by name. It contains the information required to match **attributes** in the request **context**. See Section 7.2.4.
- 2583 It also contains information to control behaviour in the event that no matching *attribute* is present in the *context*.
- 2585 Elements of this type SHALL NOT alter the match semantics of *named attributes*, but MAY narrow 2586 the search space.

```
2587
       <xs:complexType name="AttributeDesignatorType">
2588
          <xs:complexContent>
2589
             <xs:extension base="xacml:ExpressionType">
               <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
2590
2591
                <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
2592
                <xs:attribute name="Issuer" type="xs:string" use="optional"/>
2593
                <xs:attribute name="MustBePresent" type="xs:boolean" use="optional"</pre>
2594
       default="false"/>
2595
             </xs:extension>
2596
          </xs:complexContent>
2597
       </xs:complexType>
```

- A *named attribute* SHALL match an *attribute* if the values of their respective AttributeId, DataType and Issuer attributes match. The *attribute* designator's AttributeId MUST match, by URI equality, the AttributeId of the *attribute*. The *attribute* designator's DataType MUST match, by URI equality, the DataType of the same *attribute*.
- If the Issuer attribute is present in the *attribute* designator, then it MUST match, using the
 "urn:oasis:names:tc:xacml:1.0:function:string-equal" function, the Issuer of the same *attribute*. If
 the Issuer is not present in the *attribute* designator, then the matching of the *attribute* to the
 named attribute SHALL be governed by AttributeId and DataType attributes alone.

```
2606
              2607
              AttributeId [Required]
2608
                          This attribute SHALL specify the AttributeId with which to match the attribute.
2609
              DataType [Required]
2610
                          The bag returned by the <a href="https://www.energy.com/started-bag-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-returned-by-the-sage-retu
2611
                          data-type.
2612
              Issuer [Optional]
2613
                          This attribute, if supplied, SHALL specify the Issuer with which to match the attribute.
2614
              MustBePresent [Optional]
2615
                          This attribute governs whether the element returns "Indeterminate" or an empty bag in the
                          event the named attribute is absent from the request context. See Section 7.2.5. Also
2616
                          see Sections 7.15.2 and 7.15.3.
2617
                    5.38. Element <SubjectAttributeDesignator>
2618
2619
              The <SubjectAttributeDesignator> element retrieves a bag of values for a named
2620
              categorized subject attribute from the request context. A subject attribute is an attribute
              contained within a <Subject> element of the request context. A categorized subject is a subject
2621
              that is identified by a particular subject-category attribute. A named categorized subject attribute
2622
2623
              is a named subject attribute for a particular categorized subject.
2624
              The <SubjectAttributeDesignator> element SHALL return a bag containing all the subject
2625
              attribute values that are matched by the named categorized subject attribute. In the event that
2626
              no matching attribute is present in the context, the MustBePresent attribute governs whether this
2627
              element returns an empty bag or "Indeterminate". See Section 7.2.5.
2628
              The SubjectAttributeDesignatorType extends the match semantics of the
              AttributeDesignatorType (See Section 5.37) such that it narrows the attribute search space to
2629
2630
              the specific categorized subject such that the value of this element's SubjectCategory attribute
              matches, by URI equality, the value of the request context's <Subject> element's
2631
2632
              SubjectCategory attribute.
2633
              If the request context contains multiple subjects with the same SubjectCategory XML attribute,
2634
              then they SHALL be treated as if they were one categorized subject.
2635
              The <SubjectAttributeDesignator> MAY appear in the <SubjectMatch> element and
2636
              MAY be passed to the <Apply> element as an argument.
2637
              <xs:element name="SubjectAttributeDesignator"</pre>
2638
              type="xacml:SubjectAttributeDesignatorType"
2639
              substitutionGroup="xacml:Expression"/>
2640
              <xs:complexType name="SubjectAttributeDesignatorType">
2641
                   <xs:complexContent>
2642
                        <xs:extension base="xacml:AttributeDesignatorType">
2643
                             <xs:attribute name="SubjectCategory" type="xs:anyURI" use="optional"</pre>
2644
             default="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>
2645
                        </xs:extension>
2646
                   </xs:complexContent>
2647
              </xs:complexType>
```

2648 2649 2650	The <subjectattributedesignator> element is of type SubjectAttributeDesignatorType. The SubjectAttributeDesignatorType complex type extends the AttributeDesignatorType complex type with a SubjectCategory attribute.</subjectattributedesignator>
2651	SubjectCategory [Optional]
2652 2653 2654 2655	This attribute SHALL specify the <i>categorized subject</i> from which to match <i>named subject attributes</i> . If SubjectCategory is not present, then its default value of "urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" SHALL be used. Standard values of the SubjectCategory are listed in Section B.2.
2656	5.39. Element <resourceattributedesignator></resourceattributedesignator>
2657 2658 2659 2660 2661 2662	The <resourceattributedesignator> element retrieves a bag of values for a named resource attribute from the request context. A resource attribute is an attribute contained within the <resource> element of the request context. A named resource attribute is a named attribute that matches a resource attribute. A named resource attribute SHALL be considered present if there is at least one resource attribute that matches the criteria set out below. A resource attribute value is an attribute value that is contained within a resource attribute.</resource></resourceattributedesignator>
2663 2664 2665 2666	The <resourceattributedesignator> element SHALL return a bag containing all the resource attribute values that are matched by the named resource attribute. In the event that no matching attribute is present in the context, the MustBePresent attribute governs whether this element returns an empty bag or "Indeterminate". See Section 7.2.5.</resourceattributedesignator>
2667 2668	A named resource attribute SHALL match a resource attribute as per the match semantics specified in the AttributeDesignatorType complex type. See Section 5.37.
2669 2670	The <resourceattributedesignator> MAY appear in the <resourcematch> element and MAY be passed to the <apply> element as an argument.</apply></resourcematch></resourceattributedesignator>
2671 2672	<pre><xs:element name="ResourceAttributeDesignator" substitutiongroup="xacml:Expression" type="xacml:AttributeDesignatorType"></xs:element></pre>
2673 2674	The $<$ ResourceAttributeDesignator> element is of the AttributeDesignatorType complex type.
2675	5.40. Element <actionattributedesignator></actionattributedesignator>
2676 2677 2678 2679 2680 2681	The <actionattributedesignator> element retrieves a bag of values for a named action attribute from the request context. An action attribute is an attribute contained within the <action> element of the request context. A named action attribute has specific criteria (described below) with which to match an action attribute. A named action attribute SHALL be considered present, if there is at least one action attribute that matches the criteria. An action attribute value is an attribute value that is contained within an action attribute.</action></actionattributedesignator>
2682 2683 2684 2685	The <actionattributedesignator> element SHALL return a bag of all the action attribute values that are matched by the named action attribute. In the event that no matching attribute is present in the context, the MustBePresent attribute governs whether this element returns an empty bag or "Indeterminate". See Section 7.2.5.</actionattributedesignator>
2686 2687	A <i>named action attribute</i> SHALL match an <i>action attribute</i> as per the match semantics specified in the AttributeDesignatorType complex type. See Section 5.37.
2688	The <actionattributedesignator> MAY appear in the <actionmatch> element and MAY</actionmatch></actionattributedesignator>

be passed to the <Apply> element as an argument.

2689

 ${\tt 2692} \qquad {\tt The} \verb| < Action Attribute Designator > element is of the {\tt Attribute Designator Type} complex$

2693 type.

2694

5.41. Element < Environment Attribute Designator >

The <EnvironmentAttributeDesignator> element retrieves a *bag* of values for a *named*environment attribute from the request *context*. An *environment attribute* is an attribute
contained within the <Environment> element of request *context*. A named *environment*attribute has specific criteria (described below) with which to match an *environment attribute*. A
named *environment attribute* SHALL be considered *present*, if there is at least one *environment*attribute that matches the criteria. An *environment attribute* value is an *attribute* value that is

2701 contained within an *environment attribute*.

The <EnvironmentAttributeDesignator> element SHALL evaluate to a *bag* of all the
environment attribute values that are matched by the named environment attribute. In the
event that no matching attribute is present in the context, the MustBePresent attribute governs

whether this element returns an empty **bag** or "Indeterminate". See Section 7.2.5.

A named **environment attribute** SHALL match an **environment attribute** as per the match semantics specified in the **AttributeDesignatorType** complex type. See Section 5.37.

2709 argument.

2712 The <EnvironmentAttributeDesignator> element is of the AttributeDesignatorType

complex type.

2714

2717

2718

2719

2720

5.42. Element < Attribute Selector>

The <attributeSelector> element identifies attributes by their location in the request *context*.

2716 Support for the Support for the AttributeSelector> element is OPTIONAL.

The <attributeSelector> element's RequestContextPath XML attribute SHALL contain a legal XPath expression whose context node is the <acml-context:Request> element. The AttributeSelector element SHALL evaluate to a bag of values whose data-type is specified by

the element's DataType attribute. If the DataType specified in the AttributeSelector is a

primitive data type defined in [XF] or [XS], then the value returned by the XPath expression SHALL be converted to the DataType specified in the <AttributeSelector> using the constructor

function below [XF Section 4] that corresponds to the DataType. If an error results from using the constructor function, then the value of the AttributeSelector> SHALL be "Indeterminate".

2725 2726 xs:string()

2727 xs:boolean() 2728 xs:integer()

2729 xs:double()

2730 xs:dateTime() 2731 xs:date()

2732 xs:time()

2733 xs:hexBinary() 2734 xs:base64Binary()

2735 xs:anyURI()

```
2736
             xf:yearMonthDuration()
2737
             xf:dayTimeDuration()
```

2740

2741

If the DataType specified in the AttributeSelector is not one of the preceding primitive DataTypes, then the AttributeSelector SHALL return a bag of instances of the specified DataType. If an error occurs when converting the values returned by the XPath expression to the specified DataType, then the result of the AttributeSelector SHALL be "Indeterminate".

2742 2743 2744

2745 2746

2747

Each node selected by the specified XPath expression MUST be either a text node, an attribute node, a processing instruction node or a comment node. The string representation of the value of each node MUST be converted to an attribute value of the specified data-type, and the result of the AttributeSelector is the **bag** of the **attribute** values generated from all the selected nodes.

2748 2749 2750

2751

2752

If the node selected by the specified XPath expression is not one of those listed above (i.e. a text node, an attribute node, a processing instruction node or a comment node), then the result of the enclosing policy SHALL be "Indeterminate" with a StatusCode value of "urn:oasis:names:tc:xacml:1.0:status:syntax-error".

2753 2754

```
2755
       <xs:element name="AttributeSelector" type="xacml:AttributeSelectorType"</pre>
2756
       substitutionGroup="xacml:Expression"/>
2757
       <xs:complexType name="AttributeSelectorType">
2758
          <xs:complexContent>
2759
             <xs:extension base="xacml:ExpressionType">
2760
                <xs:attribute name="RequestContextPath" type="xs:string" use="required"/>
2761
                <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
2762
                <xs:attribute name="MustBePresent" type="xs:boolean" use="optional"</pre>
2763
       default="false"/>
2764
             </xs:extension>
2765
          </xs:complexContent>
2766
       </xs:complexType>
```

2767

2779

- The <attributeSelector> element is of AttributeSelectorType complex type.
- 2768 The <attributeSelector> element has the following attributes:
- 2769 RequestContextPath [Required]
- 2770 There SHALL be no restriction on the XPath syntax. See also Section 5.4. 2771
- 2772 DataType [Required]
- 2773 The bag returned by the <attributeSelector> element SHALL contain values of this 2774 data-type.
- 2775 MustBePresent [Optional]
- 2776 This attribute governs whether the element returns "Indeterminate" or an empty bag in the event the XPath expression selects no node. See Section 7.2.5. Also see Sections 7.15.2 2777 and 7.15.3. 2778

5.43. Element < Attribute Value >

2780

```
2781
        <xs:element name="AttributeValue" type="xacml:AttributeValueType"</pre>
2782
        substitutionGroup="xacml:Expression"/>
2783
        <xs:complexType name="AttributeValueType" mixed="true">
```

```
2784
          <xs:complexContent>
2785
             <xs:extension base="xacml:ExpressionType">
2786
                <xs:sequence>
2787
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"</pre>
2788
       maxOccurs="unbounded"/>
2789
               </xs:sequence>
2790
                <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
2791
                <xs:anyAttribute namespace="##any" processContents="lax"/>
2792
             </xs:extension>
2793
          </xs:complexContent>
2794
       </xs:complexType>
```

- 2797 DataType [Required]

2812

2813

2814

2815

2798 The data-type of the *attribute* value.

5.44. Element <Obligations>

- 2800 The <Obligations> element SHALL contain a set of <Obligation> elements.
- 2801 Support for the <Obligations> element is OPTIONAL.

- 2808 The <Obligations> element is of **ObligationsType** complexType.
- 2809 The <Obligations> element contains the following element:
- 2810 <Obligation> [One to Many]
- A sequence of *obligations*. See Section 5.45.

5.45. Element < Obligation>

The <Obligation> element SHALL contain an identifier for the *obligation* and a set of *attributes* that form arguments of the action defined by the *obligation*. The Fulfillon attribute SHALL indicate the *effect* for which this *obligation* must be fulfilled by the *PEP*.

```
2816
       <xs:element name="Obligation" type="xacml:ObligationType"/>
2817
       <xs:complexType name="ObligationType">
2818
          <xs:sequence>
2819
             <xs:element ref="xacml:AttributeAssignment" minOccurs="0"</pre>
2820
       maxOccurs="unbounded"/>
2821
          </xs:sequence>
2822
          <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
2823
          <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>
2824
```

- The <Obligation> element is of **ObligationType** complexType. See Section 7.14 for a description of how the set of *obligations* to be returned by the *PDP* is determined.
- 2827 The <Obligation> element contains the following elements and attributes:

```
2828
        ObligationId [Required]
2829
               Obligation identifier. The value of the obligation identifier SHALL be interpreted by the
               PEP.
2830
2831
        Fulfillon [Required]
2832
               The effect for which this obligation must be fulfilled by the PEP.
2833
        <a href="#"><AttributeAssignment>[Optional]</a>
               Obligation arguments assignment. The values of the obligation arguments SHALL be
2834
2835
               interpreted by the PEP.
            5.46. Element < Attribute Assignment >
2836
2837
        The <attributeAssignment> element is used for including arguments in obligations. It SHALL
2838
        contain an AttributeId and the corresponding attribute value, by extending the
2839
        AttributeValueType type definition. The <attributeAssignment> element MAY be used in
        any way that is consistent with the schema syntax, which is a sequence of <xs:any> elements.
2840
2841
        The value specified SHALL be understood by the PEP. but it is not further specified by XACML.
2842
        See Section 7.14. Section 4.2.4.3 provides a number of examples of arguments included in
        obligations.
2843
2844
        <xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
2845
        <xs:complexType name="AttributeAssignmentType" mixed="true">
2846
           <xs:complexContent>
2847
              <xs:extension base="xacml:AttributeValueType">
2848
                 <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
2849
              </xs:extension>
2850
           </xs:complexContent>
2851
        </xs:complexType>
2852
        The <attributeAssignment> element is of AttributeAssignmentType complex type.
2853
        The <AttributeAssignment> element contains the following attributes:
2854
        AttributeId [Required]
               The attribute Identifier.
2855
```

6. Context syntax (normative with the exception of the schema fragments)

6.1. Element < Request>

2856

2857

2858

2859

2860

2861

2862

2863 2864

2865

The <Request> element is a top-level element in the XACML *context* schema. The <Request> element is an abstraction layer used by the policy language. For simplicity of expression, this document describes *policy* evaluation in terms of operations on the *context*. However a conforming *PDP* is not required to actually instantiate the *context* in the form of an XML document. But, any system conforming to the XACML specification MUST produce exactly the same *authorization decisions* as if all the inputs had been transformed into the form of an <xacml-context:Request> element.

```
2866
               The <Request> element contains <Subject>, <Resource>, <Action> and <Environment>
               elements. There may be multiple <Subject> elements and, under some conditions, multiple
2867
               <Resource> elements<sup>2</sup>. Each child element contains a sequence of <xacml-
2868
2869
               context: Attribute > elements associated with the subject, resource, action and
2870
               environment respectively. These <a tribute> elements MAY form a part of policy evaluation.
2871
               <xs:element name="Request" type="xacml-context:RequestType"/>
2872
               <xs:complexType name="RequestType">
2873
                     <xs:sequence>
2874
                          <xs:element ref="xacml-context:Subject" maxOccurs="unbounded"/>
2875
                          <xs:element ref="xacml-context:Resource" maxOccurs="unbounded"/>
2876
                          <xs:element ref="xacml-context:Action"/>
2877
                          <xs:element ref="xacml-context:Environment"/>
2878
                     </xs:sequence>
2879
               </xs:complexType>
2880
               The <Request> element is of RequestType complex type.
2881
               The <Request> element contains the following elements:
2882
               <Subject> [One to Many]
2883
                             Specifies information about a subject of the request context by listing a sequence of
2884
                             <a href="<a 
                             are allowed. A subject is an entity associated with the access request. For example, one
2885
                             subject might represent the human user that initiated the application from which the
2886
                             request was issued; another subject might represent the application's executable code
2887
2888
                             responsible for creating the request; another subject might represent the machine on
                             which the application was executing; and another subject might represent the entity that is
2889
2890
                            to be the recipient of the resource. Attributes of each of these entities MUST be enclosed
2891
                             in separate <Subject> elements.
2892
               <Resource> [One to Many]
2893
                             Specifies information about the resource or resources for which access is being
2894
                             requested by listing a sequence of <attribute> elements associated with the resource.
                             It MAY include a <ResourceContent> element.
2895
2896
               <Action> [Required]
                             Specifies the requested action to be performed on the resource by listing a set of
2897
2898
                             <a href="#"><Attribute> elements associated with the action</a>.
2899
               <Environment> [Required]
2900
                             Contains a set of <a href="#"><a href="#">Attribute</a> elements for the environment.
                      6.2.
                                   Element <Subject>
2901
               The <Subject> element specifies a subject by listing a sequence of <Attribute> elements
2902
```

² The conditions under which multiple <Resource> elements are allowed are described in the XACML Profile for Multiple Resources [MULT].

<xs:element name="Subject" type="xacml-context:SubjectType"/>

access_control-xacml-2.0-core-spec-os

associated with the subject.

<xs:sequence>

2903

2904

2905

2906

1 February 2005

<xs:complexType name="SubjectType">

- 2913 The <Subject> element is of **SubjectType** complex type.
- 2914 The <Subject> element contains the following elements and attributes:
- 2915 SubjectCategory [Optional]

This attribute indicates the role that the parent <Subject> played in the formation of the access request. If this attribute is not present in a given <Subject> element, then the default value of "urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" SHALL be used, indicating that the parent <Subject> element represents the entity ultimately responsible for initiating the access request.

If more than one <Subject> element contains a "urn:oasis:names:tc:xacml:2.0:subject-category" attribute with the same value, then the PDP SHALL treat the contents of those elements as if they were contained in the same <Subject> element.

2924 <Attribute> [Any Number]

2921

2922

2923

2929

2930

2931

2932

2941

- 2925 A sequence of *attributes* that apply to the subject.
- Typically, a <Subject> element will contain an <Attribute> with an AttributeId of "urn:oasis:names:tc:xacml:1.0:subject:subject-id", containing the identity of the *subject*.
- 2928 A <Subject> element MAY contain additional <Attribute> elements.

6.3. Element < Resource >

The <Resource> element specifies information about the **resource** to which **access** is requested, by listing a sequence of <Attribute> elements associated with the **resource**. It MAY include the **resource** content.

```
2933
       <xs:element name="Resource" type="xacml-context:ResourceType"/>
2934
       <xs:complexType name="ResourceType">
2935
          <xs:sequence>
2936
             <xs:element ref="xacml-context:ResourceContent" minOccurs="0"/>
2937
             <xs:element ref="xacml-context:Attribute" minOccurs="0"</pre>
2938
       maxOccurs="unbounded"/>
2939
          </xs:sequence>
2940
       </xs:complexType>
```

- The <Resource> element is of ResourceType complex type.
- 2942 The <Resource> element contains the following elements:
- 2943 <ResourceContent> [Optional]
- 2944 The *resource* content.
- 2945 <Attribute> [Any Number]
- 2946 A sequence of *resource attributes*.
- The <Resource> element MAY contain one or more <Attribute> elements with an AttributeId of "urn:oasis:names:tc:xacml:2.0:resource:resource-id". Each such

2949 <a href="<"><Attribute SHALL be an absolute and fully-resolved representation of the identity of the single *resource* to which access is being requested. If there is more than one such 2950 absolute and fully-resolved representation, and if any <attribute> with this 2951 2952 AttributeId is specified, then an <Attribute> for each such distinct representation of 2953 the resource identity SHALL be specified. All such https://www.shall.nefer.com/ 2954 to the same single *resource* instance. A Profile for a particular *resource* MAY specify a 2955 single normative representation for instances of the *resource*; in this case, any <Attribute> with this AttributeId SHALL use only this one representation. 2956

A <Resource> element MAY contain additional <Attribute> elements.

6.4. Element < Resource Content>

The <ResourceContent> element is a notional placeholder for the content of the **resource**. If an XACML **policy** references the contents of the **resource** by means of an <AttributeSelector> element, then the <ResourceContent> element MUST be included in the RequestContextPath string.

- 2970 The <ResourceContent> element is of ResourceContentType complex type.
- 2971 The <ResourceContent> element allows arbitrary elements and attributes.

6.5. Element < Action>

The <action> element specifies the requested *action* on the *resource*, by listing a set of <active color of a set of <active color of action. Attribute> elements associated with the *action*.

- 2982 The <action> element is of ActionType complex type.
- 2983 The <Action> element contains the following elements:
- 2984 <Attribute> [Any Number]

2957

2958

2972

2986

2985 List of *attributes* of the *action* to be performed on the *resource*.

6.6. Element < Environment>

2987 The <Environment> element contains a set of attributes of the environment.

- 2995 The <Environment> element is of **EnvironmentType** complex type.
- 2996 The <Environment> element contains the following elements:
- 2997 <Attribute> [Any Number]

3002

3003 3004

3005

A list of *environment attributes*. Environment *attributes* are *attributes* that are not associated with either the *resource*, the *action* or any of the *subjects* of the *access* request.

6.7. Element < Attribute>

The Attribute element is the central abstraction of the request *context*. It contains *attribute* meta-data and one or more *attribute* values. The *attribute* meta-data comprises the *attribute* identifier and the *attribute* issuer. AttributeDesignator and AttributeSelector elements in the *policy* MAY refer to *attributes* by means of this meta-data.

```
3006
       <xs:element name="Attribute" type="xacml-context:AttributeType"/>
3007
       <xs:complexType name="AttributeType">
3008
          <xs:sequence>
3009
             <xs:element ref="xacml-context:AttributeValue" maxOccurs="unbounded"/>
3010
          </xs:sequence>
3011
          <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
3012
          <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
3013
          <xs:attribute name="Issuer" type="xs:string" use="optional"/>
3014
       </xs:complexType>
```

- 3015 The <a tribute > element is of AttributeType complex type.
- 3016 The Attribute element contains the following attributes and elements:
- 3017 AttributeId [Required]
- The *Attribute* identifier. A number of identifiers are reserved by XACML to denote commonly used *attributes*. See Appendix B.
- 3020 DataType [Required]

3025 Issuer [Optional]

3026

3027

3028

3030

3031

The *Attribute* issuer. For example, this attribute value MAY be an x500Name that binds to a public key, or it may be some other identifier exchanged out-of-band by issuing and relying parties.

3029 <xacml-context:AttributeValue> [One to Many]

One or more *attribute* values. Each *attribute* value MAY have contents that are empty, occur once or occur multiple times.

6.8. Element < Attribute Value >

3032

3045

3046

3047 3048

3049

3064

3065

3066

3067

3068

```
3034
       <xs:element name="AttributeValue" type="xacml-context:AttributeValueType"/>
3035
       <xs:complexType name="AttributeValueType" mixed="true">
3036
          <xs:sequence>
3037
             <xs:any namespace="##any" processContents="lax" minOccurs="0"</pre>
3038
       maxOccurs="unbounded"/>
3039
          </xs:sequence>
3040
          <xs:anyAttribute namespace="##any" processContents="lax"/>
3041
       </xs:complexType>
```

- The <pr

6.9. Element < Response>

- The <Response> element is a top-level element in the XACML *context* schema. The <Response> element is an abstraction layer used by the *policy* language. Any proprietary system using the XACML specification MUST transform an XACML *context* <Response> element into the form of its *authorization decision*.
- The <Response> element encapsulates the *authorization decision* produced by the *PDP*. It includes a sequence of one or more results, with one <Result> element per requested *resource*. Multiple results MAY be returned by some implementations, in particular those that support the XACML Profile for Requests for Multiple Resources [MULT]. Support for multiple results is OPTIONAL.

- 3060 The <Response> element is of ResponseType complex type.
- 3061 The <Response> element contains the following elements:
- 3062 <Result> [One to Many]
- 3063 An authorization decision result. See Section 6.10.

6.10. Element <Result>

The <Result> element represents an *authorization decision* result for the *resource* specified by the ResourceId *attribute*. It MAY include a set of *obligations* that MUST be fulfilled by the *PEP*. If the *PEP* does not understand or cannot fulfill an *obligation*, then it MUST act as if the *PDP* had denied *access* to the requested *resource*.

```
3076
           <xs:attribute name="ResourceId" type="xs:string" use="optional"/>
3077
        </xs:complexType>
        The <Result> element is of ResultType complex type.
3078
3079
        The <Result> element contains the following attributes and elements:
3080
        ResourceId [Optional]
3081
                The identifier of the requested resource. If this attribute is omitted, then the resource
3082
                identity is that specified by the "urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                resource attribute in the corresponding <Request> element.
3083
3084
        <Decision> [Required]
3085
                The authorization decision: "Permit", "Deny", "Indeterminate" or "NotApplicable".
3086
        <Status> [Optional]
3087
                Indicates whether errors occurred during evaluation of the decision request, and
3088
               optionally, information about those errors. If the <Response> element contains <Result>
3089
                elements whose <Status> elements are all identical, and the <Response> element is
3090
               contained in a protocol wrapper that can convey status information, then the common
3091
               status information MAY be placed in the protocol wrapper and this <Status> element
3092
                MAY be omitted from all <Result> elements.
3093
        <Obligations> [Optional]
3094
               A list of obligations that MUST be fulfilled by the PEP. If the PEP does not understand or
               cannot fulfill an obligation, then it MUST act as if the PDP had denied access to the
3095
3096
                requested resource. See Section 7.14 for a description of how the set of obligations to
               be returned by the PDP is determined.
3097
            6.11. Element < Decision>
3098
3099
        The <Decision> element contains the result of policy evaluation.
3100
        <xs:element name="Decision" type="xacml-context:DecisionType"/>
3101
        <xs:simpleType name="DecisionType">
3102
           <xs:restriction base="xs:string">
3103
              <xs:enumeration value="Permit"/>
3104
              <xs:enumeration value="Deny"/>
3105
              <xs:enumeration value="Indeterminate"/>
3106
              <xs:enumeration value="NotApplicable"/>
3107
           </xs:restriction>
3108
        </xs:simpleType>
```

- 3109 The CDecision> element is of DecisionType simple type.
- 3110 The values of the cision> element have the following meanings:
- 3111 "Permit": the requested *access* is permitted.
- 3112 "Deny": the requested *access* is denied.
- 3113 "Indeterminate": the PDP is unable to evaluate the requested *access*. Reasons for such inability include: missing *attributes*, network errors while retrieving *policies*, division by zero during *policy* evaluation, syntax errors in the *decision request* or in the *policy*, etc..
- 3116 "NotApplicable": the **PDP** does not have any **policy** that applies to this **decision request**.

3117 **6.12. Element <Status>**

3118 The <Status> element represents the status of the *authorization decision* result.

```
3119
       <xs:element name="Status" type="xacml-context:StatusType"/>
3120
       <xs:complexType name="StatusType">
3121
          <xs:sequence>
3122
            <xs:element ref="xacml-context:StatusCode"/>
3123
             <xs:element ref="xacml-context:StatusMessage" minOccurs="0"/>
3124
             <xs:element ref="xacml-context:StatusDetail" minOccurs="0"/>
3125
          </xs:sequence>
3126
       </xs:complexType>
```

- 3127 The <Status> element is of **StatusType** complex type.
- 3128 The <Status> element contains the following elements:
- 3129 <StatusCode> [Required]
- 3130 Status code.

3135

3151

- 3131 <StatusMessage> [Optional]
- 3132 A status message describing the status code.
- 3133 <StatusDetail>[Optional]
- 3134 Additional status information.

6.13. Element <StatusCode>

The <StatusCode> element contains a major status code value and an optional sequence of minor status codes.

- 3145 The <StatusCode> element is of **StatusCodeType** complex type.
- 3146 The <StatusCode> element contains the following attributes and elements:
- 3147 Value [Required]
- 3148 See Section B.9 for a list of values.
- 3149 <StatusCode> [Any Number]
- 3150 Minor status code. This status code qualifies its parent status code.

6.14. Element <StatusMessage>

- 3152 The <StatusMessage> element is a free-form description of the status code.
- 3153 <xs:element name="StatusMessage" type="xs:string"/>
- The <StatusMessage> element is of xs:string type.

6.15. Element <StatusDetail>

```
3156 The <StatusDetail> element qualifies the <Status> element with additional information.
```

- 3164 The <StatusDetail> element is of **StatusDetailType** complex type.
- 3165 The <StatusDetail> element allows arbitrary XML content.
- 3166 Inclusion of a <StatusDetail> element is optional. However, if a PDP returns one of the
- 3167 following XACML-defined <StatusCode> values and includes a <StatusDetail> element, then
- 3168 the following rules apply.

3155

- 3169 urn:oasis:names:tc:xacml:1.0:status:ok
- 3170 A *PDP* MUST NOT return a <StatusDetail> element in conjunction with the "ok" status value.
- 3171 urn:oasis:names:tc:xacml:1.0:status:missing-attribute
- 3172 A PDP MAY choose not to return any <StatusDetail> information or MAY choose to return a
- 3173 <StatusDetail> element containing one or more <xacml-context:
- 3174 MissingAttributeDetail> elements.
- 3175 urn:oasis:names:tc:xacml:1.0:status:syntax-error
- 3176 A *PDP* MUST NOT return a <StatusDetail> element in conjunction with the "syntax-error" status
- 3177 value. A syntax error may represent either a problem with the *policy* being used or with the
- 3178 request *context*. The *PDP* MAY return a <StatusMessage> describing the problem.
- 3179 urn:oasis:names:tc:xacml:1.0:status:processing-error
- 3180 A *PDP* MUST NOT return <StatusDetail> element in conjunction with the "processing-error"
- 3181 status value. This status code indicates an internal problem in the *PDP*. For security reasons, the
- 3182 **PDP** MAY choose to return no further information to the **PEP**. In the case of a divide-by-zero error
- 3183 or other computational error, the **PDP** MAY return a <StatusMessage> describing the nature of
- 3184 the error.

3185

3186

3187

6.16. Element <MissingAttributeDetail>

The <MissingAttributeDetail> element conveys information about **attributes** required for **policy** evaluation that were missing from the request **context**.

```
3188
       <xs:element name="MissingAttributeDetail" type="xacml-</pre>
3189
       context:MissingAttributeDetailType"/>
3190
       <xs:complexType name="MissingAttributeDetailType">
3191
       <xs:sequence>
3192
             <xs:element ref="xacml-context:AttributeValue" minOccurs="0"</pre>
3193
          maxOccurs="unbounded"/>
       </xs:sequence>
3194
3195
          <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
3196
          <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
3197
          <xs:attribute name="Issuer" type="xs:string" use="optional"/>
3198
       </xs:complexType>
```

3199	The <missingattributedetail> element is of MissingAttributeDetailType complex type.</missingattributedetail>
3200	The <missingattributedetal> element contains the following attributes and elements:</missingattributedetal>
3201	AttributeValue [Optional]
3202	The required value of the missing attribute.
3203	AttributeId [Required]
3204	The identifier of the missing attribute.
3205	<pre><datatype> [Required]</datatype></pre>
3206	The data-type of the missing attribute.
3207	Issuer [Optional]
3208	This attribute, if supplied, SHALL specify the required Issuer of the missing attribute.
3209 3210 3211 3212 3213 3214	If the PDP includes <xacml-context:attributevalue> elements in the <missingattributedetail> element, then this indicates the acceptable values for that attribute. If no <xacml-context:attributevalue> elements are included, then this indicates the names of attributes that the PDP failed to resolve during its evaluation. The list of attributes may be partial or complete. There is no guarantee by the PDP that supplying the missing values or attributes will be sufficient to satisfy the policy.</xacml-context:attributevalue></missingattributedetail></xacml-context:attributevalue>
3215	7. Functional requirements (normative)
3216	7. Functional requirements (normative) This section specifies certain functional requirements that are not directly associated with the production or consumption of a particular XACML element.
3216 3217	This section specifies certain functional requirements that are not directly associated with the
3216 3217 3218	This section specifies certain functional requirements that are not directly associated with the production or consumption of a particular XACML element.
3215 3216 3217 3218 3219 3220 3221 3222	This section specifies certain functional requirements that are not directly associated with the production or consumption of a particular XACML element. 7.1. Policy enforcement point
3216 3217 3218 3219 3220 3221 3222	This section specifies certain functional requirements that are not directly associated with the production or consumption of a particular XACML element. 7.1. Policy enforcement point This section describes the requirements for the PEP. An application functions in the role of the PEP if it guards access to a set of resources and asks the PDP for an authorization decision. The PEP MUST abide by the authorization decision as
3216 3217 3218 3219 3220 3221	This section specifies certain functional requirements that are not directly associated with the production or consumption of a particular XACML element. 7.1. Policy enforcement point This section describes the requirements for the PEP. An application functions in the role of the PEP if it guards access to a set of resources and asks the PDP for an authorization decision. The PEP MUST abide by the authorization decision as described in one of the following sub-sections
3216 3217 3218 3219 3220 3221 3222 3223 3223	This section specifies certain functional requirements that are not directly associated with the production or consumption of a particular XACML element. 7.1. Policy enforcement point This section describes the requirements for the PEP. An application functions in the role of the PEP if it guards access to a set of resources and asks the PDP for an authorization decision. The PEP MUST abide by the authorization decision as described in one of the following sub-sections 7.1.1. Base PEP If the decision is "Permit", then the PEP SHALL permit access. If obligations accompany the decision, then the PEP SHALL permit access only if it understands and it can and will discharge
3216 3217 3218 3219 3220 3221 3222 3223 3224 3225 3226 3227 3228	This section specifies certain functional requirements that are not directly associated with the production or consumption of a particular XACML element. 7.1. Policy enforcement point This section describes the requirements for the PEP. An application functions in the role of the PEP if it guards access to a set of resources and asks the PDP for an authorization decision. The PEP MUST abide by the authorization decision as described in one of the following sub-sections 7.1.1. Base PEP If the decision is "Permit", then the PEP SHALL permit access. If obligations accompany the decision, then the PEP SHALL permit access only if it understands and it can and will discharge those obligations. If the decision is "Deny", then the PEP SHALL deny access. If obligations accompany the decision, then the PEP shall deny access only if it understands, and it can and will discharge

7.1.2. Deny-biased PEP 3232 3233 If the **decision** is "Permit", then the **PEP** SHALL permit access. If **obligations** accompany the 3234 decision, then the PEP SHALL permit access only if it understands and it can and will discharge 3235 those obligations. All other decisions SHALL result in the denial of access. 3236 3237 Note: other actions, e.g. consultation of additional PDPs, reformulation/resubmission of the 3238 decision request, etc., are not prohibited. 7.1.3. Permit-biased PEP 3239 3240 If the decision is "Deny", then the PEP SHALL deny access. If obligations accompany the decision, then the PEP shall deny access only if it understands, and it can and will discharge 3241 3242 those *obligations*. 3243 All other decisions SHALL result in the permission of access. 3244 Note: other actions, e.g. consultation of additional PDPs, reformulation/resubmission of the 3245 decision request, etc., are not prohibited. 7.2. Attribute evaluation 3246 3247 Attributes are represented in the request context by the context handler, regardless of whether 3248 or not they appeared in the original decision request, and are referred to in the policy by subject, 3249 resource, action and environment attribute designators and attribute selectors. A named 3250 attribute is the term used for the criteria that the specific subject, resource, action and environment attribute designators and selectors use to refer to particular attributes in the 3251 subject, resource, action and environment elements of the request context, respectively. 3252 7.2.1. Structured attributes 3253 3254 <xacml:AttributeValue> and <xacml-context:AttributeValue> elements MAY contain 3255 an instance of a structured XML data-type, for example <ds:KeyInfo>. XACML 2.0 supports 3256 several ways for comparing the contents of such elements. 3257 1. In some cases, such elements MAY be compared using one of the XACML string functions, 3258 such as "string-regexp-match", described below. This requires that the element be given 3259 the data-type "http://www.w3.org/2001/XMLSchema#string". For example, a structured 3260 data-type that is actually a ds:KeyInfo/KeyName would appear in the Context as: 3261

<AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
 <ds:KeyName>jhibbert-key</ds:KeyName>
</AttributeValue>

In general, this method will not be adequate unless the structured data-type is quite simple.

- 2. An <AttributeSelector> element MAY be used to select the contents of a leaf subelement of the structured data-type by means of an XPath expression. That value MAY then be compared using one of the supported XACML functions appropriate for its primitive data-type. This method requires support by the *PDP* for the optional XPath expressions feature.
- 3. An AttributeSelector> element MAY be used to select any node in the structured data-type by means of an XPath expression. This node MAY then be compared using one

3262

3263

3264

3265

3266

3267

3268 3269

3270

3272 of the XPath-based functions described in Section A.3. This method requires support by 3273 the **PDP** for the optional XPath expressions and XPath functions features. 3274 See also Section 8.2. 7.2.2. Attribute bags 3275 3276 XACML defines implicit collections of its data-types. XACML refers to a collection of values that are 3277 of a single data-type as a bag. Bags of data-types are needed because selections of nodes from an XML resource or XACML request context may return more than one value. 3278 3279 The <attributeSelector> element uses an XPath expression to specify the selection of data 3280 from an XML resource. The result of an XPath expression is termed a node-set, which contains all 3281 the leaf nodes from the XML resource that match the predicate in the XPath expression. Based on 3282 the various indexing functions provided in the XPath specification, it SHALL be implied that a 3283 resultant node-set is the collection of the matching nodes. XACML also defines the 3284 <a href="<"><AttributeDesignator> element to have the same matching methodology for attributes in the 3285 XACML request *context*. 3286 The values in a bag are not ordered, and some of the values may be duplicates. There SHALL be 3287 no notion of a *bag* containing *bags*, or a *bag* containing values of differing types. I.e. a *bag* in 3288 XACML SHALL contain only values that are of the same data-type. 7.2.3. Multivalued attributes 3289 3290 If a single <attribute> element in a request context contains multiple <xacml-3291 context: AttributeValue> child elements, then the bag of values resulting from evaluation of 3292 the <Attribute> element MUST be identical to the bag of values that results from evaluating a 3293 context in which each cacml-context:AttributeValue> element appears in a separate 3294 <Attribute> element, each carrying identical meta-data. 3295 7.2.4. Attribute Matching 3296 A *named attribute* includes specific criteria with which to match *attributes* in the *context*. An 3297 attribute specifies an AttributeId and DataType, and a named attribute also specifies the 3298 Issuer. A named attribute SHALL match an attribute if the values of their respective 3299 AttributeId, DataType and optional Issuer attributes match within their particular element -3300 subject, resource, action or environment - of the context. The AttributeId of the named 3301 attribute MUST match, by URI equality, the AttributeId of the corresponding context attribute. 3302 The DataType of the named attribute MUST match, by URI equality, the DataType of the 3303 corresponding context attribute. If Issuer is supplied in the named attribute, then it MUST 3304 match, using the urn:oasis:names:tc:xacml:1.0:function:string-equal function, the 3305 Issuer of the corresponding *context attribute*. If Issuer is not supplied in the *named attribute*, 3306 then the matching of the *context attribute* to the *named attribute* SHALL be governed by 3307 AttributeId and DataType alone, regardless of the presence, absence, or actual value of 3308 Issuer in the corresponding context attribute. In the case of an attribute selector, the matching 3309 of the attribute to the named attribute SHALL be governed by the XPath expression and

3310

DataType.

7.2.5. Attribute Retrieval

3311

3327

3328

3329

3330

3331

3332

3333

3334

3335

3336

3337

3338 3339

3340

3341

3312 The **PDP** SHALL request the values of **attributes** in the request **context** from the **context handler**. 3313 The **PDP** SHALL reference the **attributes** as if they were in a physical request **context** document, 3314 but the *context handler* is responsible for obtaining and supplying the requested values by whatever means it deems appropriate. The *context handler* SHALL return the values of 3315 attributes that match the attribute designator or attribute selector and form them into a bag of 3316 values with the specified data-type. If no attributes from the request context match, then the 3317 3318 attribute SHALL be considered missing. If the attribute is missing, then MustBePresent 3319 governs whether the attribute designator or attribute selector returns an empty bag or an 3320 "Indeterminate" result. If MustBePresent is "False" (default value), then a missing attribute 3321 SHALL result in an empty bag. If MustBePresent is "True", then a missing attribute SHALL 3322 result in "Indeterminate". This "Indeterminate" result SHALL be handled in accordance with the 3323 specification of the encompassing expressions, rules, policies and policy sets. If the result is "Indeterminate", then the AttributeId, DataType and Issuer of the attribute MAY be listed in 3324 the authorization decision as described in Section 7.13. However, a PDP MAY choose not to 3325 3326 return such information for security reasons.

7.2.6. Environment Attributes

Standard *environment attributes* are listed in Section B.8. If a value for one of these *attributes* is supplied in the *decision request*, then the *context handler* SHALL use that value. Otherwise, the *context handler* SHALL supply a value. In the case of date and time *attributes*, the supplied value SHALL have the semantics of the "date and time that apply to the *decision request*".

7.3. Expression evaluation

XACML specifies expressions in terms of the elements listed below, of which the <Apply> and <Condition> elements recursively compose greater expressions. Valid expressions SHALL be type correct, which means that the types of each of the elements contained within <Apply> and <Condition> elements SHALL agree with the respective argument types of the function that is named by the FunctionId attribute. The resultant type of the <Apply> or <Condition> element SHALL be the resultant type of the function, which MAY be narrowed to a primitive datatype, or a *bag* of a primitive data-type, by type-unification. XACML defines an evaluation result of "Indeterminate", which is said to be the result of an invalid expression, or an operational error occurring during the evaluation of the expression.

- 3342 XACML defines these elements to be in the substitution group of the <Expression> element:
- 3343 <xacml:AttributeValue>
- 3344 <xacml:SubjectAttributeDesignator>
- 3345 <xacml:ResourceAttributeDesignator>
- **3346** <xacml:ActionAttributeDesignator>
- 3348 <xacml:AttributeSelector>
- **3349** < xacml : Apply>
- 3350 <xacml:Condition>

3351 <xacml:Function> 3352 <xacml:VariableReference> 7.4. Arithmetic evaluation 3353 3354 IEEE 754 [IEEE 754] specifies how to evaluate arithmetic functions in a context, which specifies defaults for precision, rounding, etc. XACML SHALL use this specification for the evaluation of all 3355 3356 integer and double functions relying on the Extended Default Context, enhanced with double 3357 precision: 3358 flags - all set to 0 3359 trap-enablers - all set to 0 (IEEE 854 §7) with the exception of the "division-by-zero" trap enabler, which SHALL be set to 1 3360 3361 precision - is set to the designated double precision 3362 rounding - is set to round-half-even (IEEE 854 §4.1) 7.5. Match evaluation 3363 3364 Attribute matching elements appear in the <Target> element of rules, policies and policy sets. They are the following: 3365 3366 <SubjectMatch> 3367 <ResourceMatch> 3368 <ActionMatch> 3369 <EnvironmentMatch> 3370 These elements represent Boolean expressions over attributes of the subject, resource, action 3371 and environment, respectively. A matching element contains a MatchId attribute that specifies 3372 the function to be used in performing the match evaluation, an <xacml:AttributeValue> and an 3373 <a href="<"><a href=" <a h 3374 context that is to be matched against the specified value. 3375 The Matchid attribute SHALL specify a function that compares two arguments, returning a result 3376 type of "http://www.w3.org/2001/XMLSchema#boolean". The attribute value specified in the 3377 matching element SHALL be supplied to the MatchId function as its first argument. An element of the bag returned by the AttributeSelector element SHALL 3378 3379 be supplied to the MatchId function as its second argument, as explained below. The DataType 3380 3381 the MatchId function. The DataType of the <AttributeDesignator> or 3382 <attributeSelector> element SHALL match the data-type of the second argument expected 3383 by the Matchid function. 3384 The XACML standard functions that meet the requirements for use as a Matchid attribute value 3385 3386 urn:oasis:names:tc:xacml:2.0:function:-type-equal 3387 urn:oasis:names:tc:xacml:2.0:function:-type-greater-than 3388 urn:oasis:names:tc:xacml:2.0:function:-type-greater-than-or-equal

```
    urn:oasis:names:tc:xacml:2.0:function:-type-less-than
    urn:oasis:names:tc:xacml:2.0:function:-type-less-than-or-equal
    urn:oasis:names:tc:xacml:2.0:function:-type-match
```

In addition, functions that are strictly within an extension to XACML MAY appear as a value for the Matchid attribute, and those functions MAY use data-types that are also extensions, so long as the extension function returns a Boolean result and takes two single base types as its inputs. The function used as the value for the Matchid attribute SHOULD be easily indexable. Use of non-indexable or complex functions may prevent efficient evaluation of *decision requests*.

The evaluation semantics for a matching element is as follows. If an operational error were to occur while evaluating the AttributeSelector element, then the result of the entire expression SHALL be "Indeterminate". If the AttributeDesignator or AttributeSelector element were to evaluate to an empty <code>bag</code>, then the result of the expression SHALL be "False". Otherwise, the <code>Matchld</code> function SHALL be applied between the AttributeValue and each element of the <code>bag</code> returned from the AttributeDesignator or AttributeSelector element. If at least one of those function applications were to evaluate to "True", then the result of the entire expression SHALL be "Indeterminate", then the result of the entire expression SHALL be "False".

It is also possible to express the semantics of a *target* matching element in a *condition*. For instance, the *target* match expression that compares a "subject-name" starting with the name "John" can be expressed as follows:

Alternatively, the same match semantics can be expressed as an <Apply> element in a *condition* by using the "urn:oasis:names:tc:xacml:1.0:function:any-of" function, as follows:

7.6. Target evaluation

The *target* value SHALL be "Match" if the *subjects*, *resources*, *actions* and *environments* specified in the *target* all match values in the request *context*. If any one of the *subjects*, *resources*, *actions* and *environments* specified in the *target* are "Indeterminate", then the *target* SHALL be "Indeterminate". Otherwise, the *target* SHALL be "No match". The *target* match table is shown in Table 1.

Subjects value	Resources value	Actions value	Environments value	Target value
"Match"	"Match"	"Match"	"Match"	"Match"
"No match"	"Match" or "No match"	"Match" or "No match"	"Match" or "No match"	"No match"
"Match" or "No match"	"No match"	"Match" or "No match"	"Match" or "No match"	"No match"
"Match" or "No match"	"Match" or "No match"	"No match"	"Match" or "No match"	"No match"
"Match" or "No match"	"Match" or "No match"	"Match" or "No match"	"No match"	"No match"
"Indeterminate"	Don't care	Don't care	Don't care	"Indeterminate"
Don't care	"Indeterminate"	Don't care	Don't care	"Indeterminate"
Don't care	Don't care	"Indeterminate"	Don't care	"Indeterminate"
Don't care	Don't care	Don't care	"Indeterminate"	"Indeterminate"

Table 1 - Target match table

3439 3440 3441

3442 3443 The *subjects*, *resources*, *actions* and *environments* SHALL match values in the request *context* if at least one of their <Subject>, <Resource>, <Action> or <Environment> elements, respectively, matches a value in the request *context*. The *subjects* match table is shown in Table 2. The *resources*, *actions* and *environments* match tables are analogous.

<subject> values</subject>	<subjects> Value</subjects>
At least one "Match"	"Match"
None matches and at least one "Indeterminate"	"Indeterminate"
All "No match"	"No match"

3444 3445 Table 2 - Subjects match table

3446

3448

A **subject**, **resource**, **action** or **environment** SHALL match a value in the request **context** if the value of all its <SubjectMatch>, <ResourceMatch>, <ActionMatch> or

The *subject* match table is shown in Table 3. The *resource*, *action* and *environment* match

3447 <E

tables are analogous.

<EnvironmentMatch> elements, respectively, are "True".

3449

3450

<subjectmatch> values</subjectmatch>	<subject> Value</subject>
All "True"	"Match"
No "False" and at least one "Indeterminate"	"Indeterminate"
At least one "False"	"No match"

3454

3455

3456

3457 3458

3459

3460

3461

3462

3463

3464

3465

3470

3471

3472

3473

3477

3478

7.7. VariableReference Evaluation

The <VariableReference> element references a single <VariableDefinition> element contained within the same <Policy> element. A <VariableReference> that does not reference a particular <VariableDefinition> element within the encompassing <Policy> element is called an undefined reference. *Policies* with undefined references are invalid.

In any place where a <VariableReference> occurs, it has the effect as if the text of the <Expression> element defined in the <VariableDefinition> element replaces the <VariableReference> element. Any evaluation scheme that preserves this semantic is acceptable. For instance, the expression in the <VariableDefinition> element may be evaluated to a particular value and cached for multiple references without consequence. (I.e. the value of an <Expression> element remains the same for the entire policy evaluation.) This characteristic is one of the benefits of XACML being a declarative language.

7.8. Condition evaluation

The *condition* value SHALL be "True" if the <Condition> element is absent, or if it evaluates to "True". Its value SHALL be "False" if the <Condition> element evaluates to "False". The condition value SHALL be "Indeterminate", if the expression contained in the <Condtion> element evaluates to "Indeterminate."

7.9. Rule evaluation

A *rule* has a value that can be calculated by evaluating its contents. *Rule* evaluation involves separate evaluation of the *rule's target* and *condition*. The *rule* truth table is shown in Table 4.

Target	Condition	Rule Value
"Match"	"True"	Effect
"Match"	"False"	"NotApplicable"
"Match"	"Indeterminate"	"Indeterminate"
"No-match"	Don't care	"NotApplicable"
"Indeterminate"	Don't care	"Indeterminate"

Table 4 - Rule truth table

3474 If the *target* value is "No-match" or "Indeterminate" then the *rule* value SHALL be "NotApplicable" or "Indeterminate", respectively, regardless of the value of the *condition*. For these cases, 3476 therefore, the *condition* need not be evaluated.

If the *target* value is "Match" and the *condition* value is "True", then the *effect* specified in the enclosing <Rule> element SHALL determine the *rule's* value.

7.10. Policy evaluation 3479

3483

3484

3485 3486

3487 3488

3490

3491

3492

3493 3494

3495

3496

3497 3498

3499

3500

3501 3502

3503

3504

3505

3506 3507

3508

3509

3480 The value of a policy SHALL be determined only by its contents, considered in relation to the 3481 contents of the request context. A policy's value SHALL be determined by evaluation of the 3482 policy's target and rules.

The policy's target SHALL be evaluated to determine the applicability of the policy. If the target evaluates to "Match", then the value of the policy SHALL be determined by evaluation of the policy's rules, according to the specified rule-combining algorithm. If the target evaluates to "No-match", then the value of the *policy* SHALL be "NotApplicable". If the *target* evaluates to "Indeterminate", then the value of the *policy* SHALL be "Indeterminate".

The **policy** truth table is shown in Table 5.

Target	Rule values	Policy Value
"Match"	At least one rule value is its Effect	Specified by the <i>rule-combining algorithm</i>
"Match"	All rule values are "NotApplicable"	"NotApplicable"
"Match"	At least one rule value is "Indeterminate"	Specified by the <i>rule-combining algorithm</i>
"No-match"	Don't care	"NotApplicable"
"Indeterminate"	Don't care	"Indeterminate"

3489 Table 5 - Policy truth table

> A rules value of "At least one rule value is its Effect" means either that the <Rule> element is absent, or one or more of the *rules* contained in the *policy* is applicable to the *decision request* (i.e., it returns the value of its "Effect"; see Section 7.9). A rules value of "All rule values are 'NotApplicable'" SHALL be used if no *rule* contained in the *policy* is applicable to the request and if no rule contained in the policy returns a value of "Indeterminate". If no rule contained in the policy is applicable to the request, but one or more rule returns a value of "Indeterminate", then the rules SHALL evaluate to "At least one rule value is 'Indeterminate'".

If the target value is "No-match" or "Indeterminate" then the policy value SHALL be "NotApplicable" or "Indeterminate", respectively, regardless of the value of the rules. For these cases, therefore, the *rules* need not be evaluated.

If the target value is "Match" and the rule value is "At least one rule value is it's Effect" or "At least one rule value is 'Indeterminate'", then the *rule-combining algorithm* specified in the *policy* SHALL determine the **policy** value.

Note that none of the *rule-combining algorithms* defined by XACML 2.0 take parameters. However, non-standard *combining algorithms* MAY take parameters. In such a case, the values of these parameters associated with the rules, MUST be taken into account when evaluating the policy. The parameters and their types should be defined in the specification of the combining **algorithm.** If the implementation supports combiner parameters and if combiner parameters are present in a *policy*, then the parameter values MUST be supplied to the *combining algorithm* implementation.

7.11. Policy Set evaluation

The value of a *policy set* SHALL be determined by its contents, considered in relation to the contents of the *request context*. A *policy set's* value SHALL be determined by evaluation of the *policy set's target*, *policies* and *policy sets*, according to the specified *policy-combining* algorithm.

The *policy set's target* SHALL be evaluated to determine the applicability of the *policy set*. If the *target* evaluates to "Match" then the value of the *policy set* SHALL be determined by evaluation of the *policy set's policies* and *policy sets*, according to the specified *policy-combining algorithm*. If the *target* evaluates to "No-match", then the value of the *policy set* shall be "NotApplicable". If the *target* evaluates to "Indeterminate", then the value of the *policy set* SHALL be "Indeterminate".

The **policy set** truth table is shown in Table 6.

Target	Policy values	Policy Set Value
"Match"	At least one policy value is its Decision	Specified by the <i>policy-combining algorithm</i>
"Match"	All policy values are "NotApplicable"	"NotApplicable"
"Match"	At least one policy value is "Indeterminate"	Specified by the <i>policy-combining algorithm</i>
"No-match"	Don't care	"NotApplicable"
"Indeterminate"	Don't care	"Indeterminate"

Table 6 – Policy set truth table

A *policies* value of "At least one policy value is its *Decision*" SHALL be used if there are no contained or referenced *policies* or *policy sets*, or if one or more of the *policies* or *policy sets* contained in or referenced by the *policy set* is applicable to the *decision request* (i.e., returns a value determined by its *combining algorithm*) A *policies* value of "All policy values are 'NotApplicable'" SHALL be used if no *policy* or *policy set* contained in or referenced by the *policy set* is applicable to the request and if no *policy* or *policy set* contained in or referenced by the *policy set* returns a value of "Indeterminate". If no *policy* or *policy set* contained in or referenced by the *policy set* is applicable to the request but one or more *policy set* returns a value of "Indeterminate", then the *policies* SHALL evaluate to "At least one policy value is 'Indeterminate'".

If the *target* value is "No-match" or "Indeterminate" then the *policy set* value SHALL be
"NotApplicable" or "Indeterminate", respectively, regardless of the value of the *policies*. For these cases, therefore, the *policies* need not be evaluated.

If the *target* value is "Match" and the *policies* value is "At least one policy value is its *Decision*" or "At least one policy value is 'Indeterminate'", then the *policy-combining algorithm* specified in the *policy set* SHALL determine the *policy set* value.

Note that none of the *policy-combining algorithms* defined by XACML 2.0 take parameters. However, non-standard *combining algorithms* MAY take parameters. In such a case, the values of these parameters associated with the *policies*, MUST be taken into account when evaluating the *policy set*. The parameters and their types should be defined in the specification of the

3542 3543 3544	combining algorithm . If the implementation supports combiner parameters and if combiner parameters are present in a policy , then the parameter values MUST be supplied to the combining algorithm implementation.
3545	7.12. Hierarchical resources
3546 3547 3548 3549	It is often the case that a <i>resource</i> is organized as a hierarchy (e.g. file system, XML document). XACML provides several optional mechanisms for supporting hierarchical resources. These are described in the XACML Profile for Hierarchical Resources [HIER] and in the XACML Profile for Requests for Multiple Resources [MULT].
3550	7.13. Authorization decision
3551 3552 3553 3554	In relation to a particular <i>decision request</i> , the <i>PDP</i> is defined by a <i>policy-combining algorithm</i> and a set of <i>policies</i> and/or <i>policy sets</i> . The <i>PDP</i> SHALL return a response <i>context</i> as if it had evaluated a single <i>policy set</i> consisting of this <i>policy-combining algorithm</i> and the set of <i>policies</i> and/or <i>policy sets</i> .
3555 3556 3557	The PDP MUST evaluate the policy set as specified in Sections 5 and 7. The PDP MUST return a response context , with one <decision> element of value "Permit", "Deny", "Indeterminate" or "NotApplicable".</decision>
3558 3559	If the PDP cannot make a decision, then an "Indeterminate" <decision> element SHALL be returned.</decision>
3560	7.14. Obligations
3561 3562 3563 3564	A <i>policy</i> or <i>policy set</i> may contain one or more <i>obligations</i> . When such a <i>policy</i> or <i>policy set</i> is evaluated, an <i>obligation</i> SHALL be passed up to the next level of evaluation (the enclosing or referencing <i>policy</i> , <i>policy set</i> or <i>authorization decision</i>) only if the <i>effect</i> of the <i>policy</i> or <i>policy set</i> being evaluated matches the value of the Fulfillon attribute of the <i>obligation</i> .
3565 3566 3567 3568 3569	As a consequence of this procedure, no obligations SHALL be returned to the PEP if the policies or policy sets from which they are drawn are not evaluated, or if their evaluated result is "Indeterminate" or "NotApplicable", or if the decision resulting from evaluating the policy or policy set does not match the decision resulting from evaluating an enclosing policy set .
3570 3571 3572 3573 3574 3575	If the <i>PDP's</i> evaluation is viewed as a tree of <i>policy sets</i> and <i>policies</i> , each of which returns "Permit" or "Deny", then the set of <i>obligations</i> returned by the <i>PDP</i> to the <i>PEP</i> will include only the <i>obligations</i> associated with those paths where the <i>effect</i> at each level of evaluation is the same as the <i>effect</i> being returned by the <i>PDP</i> . In situations where any lack of determinism is unacceptable, a deterministic combining algorithm, such as ordered-deny-overrides, should be used.
3576	Also, see Section 7.1.
3577	7.15. Exception handling
3578	XACML specifies behaviour for the <i>PDP</i> in the following situations.
3579	7.15.1. Unsupported functionality
3580 3581	If the PDP attempts to evaluate a policy set or policy that contains an optional element type or function that the PDP does not support, then the PDP SHALL return a specific on a value of

3582 3583 3584	"Indeterminate". If a <statuscode> element is also returned, then its value SHALL be "urn:oasis:names:tc:xacml:1.0:status:syntax-error" in the case of an unsupported element type, and "urn:oasis:names:tc:xacml:1.0:status:processing-error" in the case of an unsupported function.</statuscode>
3585	7.15.2. Syntax and type errors
3586 3587 3588	If a policy that contains invalid syntax is evaluated by the XACML PDP at the time a decision request is received, then the result of that policy SHALL be "Indeterminate" with a StatusCode value of "urn:oasis:names:tc:xacml:1.0:status:syntax-error".
3589 3590 3591	If a <i>policy</i> that contains invalid static data-types is evaluated by the XACML <i>PDP</i> at the time a <i>decision request</i> is received, then the result of that <i>policy</i> SHALL be "Indeterminate" with a StatusCode value of "urn:oasis:names:tc:xacml:1.0:status:processing-error".
3592	7.15.3. Missing attributes
3593 3594 3595 3596	The absence of matching <i>attributes</i> in the request <i>context</i> for any of the <i>attribute</i> designators or selectors that are found in the <i>policy</i> SHALL result in a <decision> element containing the "Indeterminate" value, as described in Sections 5.37 and 5.42. If, in this case, and a status code is supplied, then the value</decision>
3597	"urn:oasis:names:tc:xacml:1.0:status:missing-attribute"
3598 3599 3600 3601 3602	SHALL be used, to indicate that more information is needed in order for a definitive decision to be rendered. In this case, the <status> element MAY list the names and data-types of any attributes of the subjects, resource, action or environment that are needed by the PDP to refine its decision (see Section 6.16). A PEP MAY resubmit a refined request context in response to a <decision> element contents of "Indeterminate" with a status code of</decision></status>
3603	"urn:oasis:names:tc:xacml:1.0:missing-attribute"
3604 3605	by adding attribute values for the attribute names that were listed in the previous response. When the PDP returns a <decision> element contents of "Indeterminate", with a status code of</decision>
3606	"urn:oasis:names:tc:xacml:1.0:missing-attribute",
3607 3608 3609 3610	it MUST NOT list the names and data-types of any <i>attribute</i> of the <i>subject</i> , <i>resource</i> , <i>action</i> or <i>environment</i> for which values were supplied in the original request. Note, this requirement forces the <i>PDP</i> to eventually return an <i>authorization decision</i> of "Permit", "Deny" or "Indeterminate" with some other status code, in response to successively-refined requests.
3611	8. XACML extensibility points (non-normative)
3612 3613	This section describes the points within the XACML model and schema where extensions can be added
3614	8.1. Extensible XML attribute types
3615 3616	The following XML attributes have values that are URIs. These may be extended by the creation of new URIs associated with new semantics for these attributes.
3617	AttributeId,

3618	DataType,
3619	FunctionId,
3620	Matchid,
3621	ObligationId,
3622	PolicyCombiningAlgId,
3623	RuleCombiningAlgId,
3624	StatusCode,
3625	SubjectCategory.
3626	See Section 5 for definitions of these attribute types.
3627	8.2. Structured attributes
3628 3629 3630 3631	<pre><xacml:attributevalue> and <xacml-context:attributevalue> elements MAY contain an instance of a structured XML data-type. Section 7.2.1 describes a number of standard techniques to identify data items within such a structured attribute. Listed here are some additional techniques that require XACML extensions.</xacml-context:attributevalue></xacml:attributevalue></pre>
3632 3633 3634 3635 3636 3637 3638 3639	1. For a given structured data-type, a community of XACML users MAY define new attribute identifiers for each leaf sub-element of the structured data-type that has a type conformant with one of the XACML-defined primitive data-types. Using these new attribute identifiers, the <i>PEPs</i> or <i>context handlers</i> used by that community of users can flatten instances of the structured data-type into a sequence of individual

3655 3656	actors upon which it relies. Mechanisms for trust establishment are outside the scope of this specification.
3657 3658 3659 3660	The messages that are transmitted between the actors in the XACML model are susceptible to attack by malicious third parties. Other points of vulnerability include the <i>PEP</i> , the <i>PDP</i> and the <i>PAP</i> . While some of these entities are not strictly within the scope of this specification, their compromise could lead to the compromise of <i>access control</i> enforced by the <i>PEP</i> .
3661 3662 3663 3664	It should be noted that there are other components of a distributed system that may be compromised, such as an operating system and the domain-name system (DNS) that are outside the scope of this discussion of threat models. Compromise in these components may also lead to a policy violation.
3665 3666	The following sections detail specific compromise scenarios that may be relevant to an XACML system.
3667	9.1.1. Unauthorized disclosure
3668 3669 3670 3671 3672 3673	XACML does not specify any inherent mechanisms to protect the confidentiality of the messages exchanged between actors. Therefore, an adversary could observe the messages in transit. Under certain security policies, disclosure of this information is a violation. Disclosure of <i>attributes</i> or the types of <i>decision requests</i> that a <i>subject</i> submits may be a breach of privacy policy. In the commercial sector, the consequences of unauthorized disclosure of personal data may range from embarrassment to the custodian to imprisonment and large fines in the case of medical or financial data.
3675	Unauthorized disclosure is addressed by confidentiality safeguards.
3676	9.1.2. Message replay
3676 3677 3678 3679	9.1.2. Message replay A message replay attack is one in which the adversary records and replays legitimate messages between XACML actors. This attack may lead to denial of service, the use of out-of-date information or impersonation.
3677 3678	A message replay attack is one in which the adversary records and replays legitimate messages between XACML actors. This attack may lead to denial of service, the use of out-of-date
3677 3678 3679	A message replay attack is one in which the adversary records and replays legitimate messages between XACML actors. This attack may lead to denial of service, the use of out-of-date information or impersonation.
3677 3678 3679 3680 3681	A message replay attack is one in which the adversary records and replays legitimate messages between XACML actors. This attack may lead to denial of service, the use of out-of-date information or impersonation. Prevention of replay attacks requires the use of message freshness safeguards. Note that encryption of the message does not mitigate a replay attack since the message is simply
3677 3678 3679 3680 3681 3682	A message replay attack is one in which the adversary records and replays legitimate messages between XACML actors. This attack may lead to denial of service, the use of out-of-date information or impersonation. Prevention of replay attacks requires the use of message freshness safeguards. Note that encryption of the message does not mitigate a replay attack since the message is simply replayed and does not have to be understood by the adversary.
3677 3678 3679 3680 3681 3682 3683	A message replay attack is one in which the adversary records and replays legitimate messages between XACML actors. This attack may lead to denial of service, the use of out-of-date information or impersonation. Prevention of replay attacks requires the use of message freshness safeguards. Note that encryption of the message does not mitigate a replay attack since the message is simply replayed and does not have to be understood by the adversary. 9.1.3. Message insertion A message insertion attack is one in which the adversary inserts messages in the sequence of
3677 3678 3679 3680 3681 3682 3683 3684 3685 3686 3687 3688 3688 3689	A message replay attack is one in which the adversary records and replays legitimate messages between XACML actors. This attack may lead to denial of service, the use of out-of-date information or impersonation. Prevention of replay attacks requires the use of message freshness safeguards. Note that encryption of the message does not mitigate a replay attack since the message is simply replayed and does not have to be understood by the adversary. 9.1.3. Message insertion A message insertion attack is one in which the adversary inserts messages in the sequence of messages between XACML actors. The solution to a message insertion attack is to use mutual authentication and message sequence integrity safeguards between the actors. It should be noted that just using SSL mutual authentication is not sufficient. This only proves that the other party is the one identified by the subject of the X.509 certificate. In order to be effective, it is necessary to confirm that the certificate

3694 3695	properly designed XACML system should not render an incorrect authorization decision as a result of a message deletion attack.
3696 3697	The solution to a message deletion attack is to use message sequence integrity safeguards between the actors.
3698	9.1.5. Message modification
3699 3700 3701	If an adversary can intercept a message and change its contents, then they may be able to alter an authorization decision. A message integrity safeguard can prevent a successful message modification attack.
3702	9.1.6. NotApplicable results
3703 3704 3705 3706	A result of "NotApplicable" means that the <i>PDP</i> could not locate a <i>policy</i> whose <i>target</i> matched the information in the <i>decision request</i> . In general, it is highly recommended that a "Deny" <i>effect policy</i> be used, so that when a <i>PDP</i> would have returned "NotApplicable", a result of "Deny" is returned instead.
3707 3708 3709 3710	In some security models, however, such as those found in many Web Servers, an authorization decision of "NotApplicable" is treated as equivalent to "Permit". There are particular security considerations that must be taken into account for this to be safe. These are explained in the following paragraphs.
3711 3712 3713 3714 3715	If "NotApplicable" is to be treated as "Permit", it is vital that the matching algorithms used by the policy to match elements in the decision request be closely aligned with the data syntax used by the applications that will be submitting the decision request . A failure to match will result in "NotApplicable" and be treated as "Permit". So an unintended failure to match may allow unintended access .
3716 3717 3718 3719 3720	Commercial http responders allow a variety of syntaxes to be treated equivalently. The "%" can be used to represent characters by hex value. The URL path "//" provides multiple ways of specifying the same value. Multiple character sets may be permitted and, in some cases, the same printed character can be represented by different binary values. Unless the matching algorithm used by the policy is sophisticated enough to catch these variations, unintended access may be permitted.
3721 3722 3723 3724 3725 3726 3727 3728	It may be safe to treat "NotApplicable" as "Permit" only in a closed environment where all applications that formulate a <i>decision request</i> can be guaranteed to use the exact syntax expected by the <i>policies</i> . In a more open environment, where <i>decision requests</i> may be received from applications that use any legal syntax, it is strongly recommended that "NotApplicable" NOT be treated as "Permit" unless matching rules have been very carefully designed to match all possible applicable inputs, regardless of syntax or type variations. Note, however, that according to Section 7.1, a <i>PEP</i> must deny <i>access</i> unless it receives an explicit "Permit" <i>authorization decision</i> .
3729	9.1.7. Negative rules
3730 3731 3732 3733 3734	A negative <i>rule</i> is one that is based on a <i>predicate</i> not being "True". If not used with care, negative <i>rules</i> can lead to a policy violation, therefore some authorities recommend that they not be used. However, negative <i>rules</i> can be extremely efficient in certain cases, so XACML has chosen to include them. Nevertheless, it is recommended that they be used with care and avoided if possible.
3735 3736 3737	A common use for negative <i>rules</i> is to deny <i>access</i> to an individual or subgroup when their membership in a larger group would otherwise permit them access. For example, we might want to write a <i>rule</i> that allows all Vice Presidents to see the unpublished financial data, except for Joe,

3738	who is only a Ceremonia	Vice President and car	be indiscreet in his	communications.	If we have
------	-------------------------	------------------------	----------------------	-----------------	------------

- 3739 complete control over the administration of *subject attributes*, a superior approach would be to
- 3740 define "Vice President" and "Ceremonial Vice President" as distinct groups and then define *rules*
- 3741 accordingly. However, in some environments this approach may not be feasible. (It is worth noting
- in passing that, generally speaking, referring to individuals in *rules* does not scale well. Generally,
- 3743 shared *attributes* are preferred.)
- 3744 If not used with care, negative *rules* can lead to policy violation in two common cases. They are:
- 3745 when attributes are suppressed and when the base group changes. An example of suppressed
- 3746 attributes would be if we have a policy that access should be permitted, unless the subject is a
- 3747 credit risk. If it is possible that the *attribute* of being a credit risk may be unknown to the *PDP* for
- 3748 some reason, then unauthorized **access** may result. In some environments, the **subject** may be
- 3749 able to suppress the publication of *attributes* by the application of privacy controls, or the server or
- 3750 repository that contains the information may be unavailable for accidental or intentional reasons.
- 3751 An example of a changing base group would be if there is a policy that everyone in the engineering
- 3752 department may change software source code, except for secretaries. Suppose now that the
- 3753 department was to merge with another engineering department and the intent is to maintain the
- 3754 same policy. However, the new department also includes individuals identified as administrative
- 3755 assistants, who ought to be treated in the same way as secretaries. Unless the policy is altered.
- 3756 they will unintentionally be permitted to change software source code. Problems of this type are
- 3757 easy to avoid when one individual administers all *policies*, but when administration is distributed,
- 3758 as XACML allows, this type of situation must be explicitly guarded against.

9.2. Safeguards

3759

3760

3774

9.2.1. Authentication

- Authentication provides the means for one party in a transaction to determine the identity of the
- other party in the transaction. Authentication may be in one direction, or it may be bilateral.
- 3763 Given the sensitive nature of *access control* systems, it is important for a *PEP* to authenticate the
- 3764 identity of the **PDP** to which it sends **decision requests**. Otherwise, there is a risk that an
- 3765 adversary could provide false or invalid *authorization decisions*, leading to a policy violation.
- 3766 It is equally important for a *PDP* to authenticate the identity of the *PEP* and assess the level of trust
- 3767 to determine what, if any, sensitive data should be passed. One should keep in mind that even
- 3768 simple "Permit" or "Deny" responses could be exploited if an adversary were allowed to make
- 3769 unlimited requests to a **PDP**.
- 3770 Many different techniques may be used to provide authentication, such as co-located code, a
- 3771 private network, a VPN or digital signatures. Authentication may also be performed as part of the
- 3772 communication protocol used to exchange the *contexts*. In this case, authentication may be
- performed either at the message level or at the session level.

9.2.2. Policy administration

- 3775 If the contents of *policies* are exposed outside of the *access control* system, potential *subjects*
- may use this information to determine how to gain unauthorized *access*.
- 3777 To prevent this threat, the repository used for the storage of *policies* may itself require *access*
- 3778 control. In addition, the <Status> element should be used to return values of missing attributes
- 3779 only when exposure of the identities of those attributes will not compromise security.

9.2.3. Confidentiality 3780 3781 Confidentiality mechanisms ensure that the contents of a message can be read only by the desired 3782 recipients and not by anyone else who encounters the message while it is in transit. There are two 3783 areas in which confidentiality should be considered: one is confidentiality during transmission; the 3784 other is confidentiality within a <Policy> element. 9.2.3.1. Communication confidentiality 3785 3786 In some environments it is deemed good practice to treat all data within an access control system 3787 as confidential. In other environments, policies may be made freely available for distribution, inspection and audit. The idea behind keeping policy information secret is to make it more difficult 3788 for an adversary to know what steps might be sufficient to obtain unauthorized access. Regardless 3789 3790 of the approach chosen, the security of the access control system should not depend on the 3791 secrecy of the policy. 3792 Any security considerations related to transmitting or exchanging XACML <Policy> elements are 3793 outside the scope of the XACML standard. While it is often important to ensure that the integrity 3794 and confidentiality of <Policy> elements is maintained when they are exchanged between two 3795 parties, it is left to the implementers to determine the appropriate mechanisms for their 3796 environment. 3797 Communications confidentiality can be provided by a confidentiality mechanism, such as SSL. 3798 Using a point-to-point scheme like SSL may lead to other vulnerabilities when one of the end-points 3799 is compromised. 9.2.3.2. Statement level confidentiality 3800 3801 In some cases, an implementation may want to encrypt only parts of an XACML <Policy> 3802 element. 3803 The XML Encryption Syntax and Processing Candidate Recommendation from W3C can be used 3804 to encrypt all or parts of an XML document. This specification is recommended for use with 3805 XACML. 3806 It should go without saying that if a repository is used to facilitate the communication of cleartext 3807 (i.e., unencrypted) policy between the PAP and PDP, then a secure repository should be used to 3808 store this sensitive data. 9.2.4. Policy integrity 3809 3810 The XACML *policy*, used by the *PDP* to evaluate the request *context*, is the heart of the system. 3811 Therefore, maintaining its integrity is essential. There are two aspects to maintaining the integrity of 3812 the **policy**. One is to ensure that <Policy> elements have not been altered since they were 3813 originally created by the PAP. The other is to ensure that <Policy> elements have not been 3814 inserted or deleted from the set of policies. 3815 In many cases, both aspects can be achieved by ensuring the integrity of the actors and 3816 implementing session-level mechanisms to secure the communication between actors. The 3817 selection of the appropriate mechanisms is left to the implementers. However, when policy is 3818 distributed between organizations to be acted on at a later time, or when the policy travels with the 3819 protected resource, it would be useful to sign the policy. In these cases, the XML Signature 3820 Syntax and Processing standard from W3C is recommended to be used with XACML.

Digital signatures should only be used to ensure the integrity of the statements. Digital signatures should not be used as a method of selecting or evaluating *policy*. That is, the *PDP* should not request a *policy* based on who signed it or whether or not it has been signed (as such a basis for selection would, itself, be a matter of policy). However, the *PDP* must verify that the key used to sign the *policy* is one controlled by the purported issuer of the *policy*. The means to do this are dependent on the specific signature technology chosen and are outside the scope of this document.

9.2.5. Policy identifiers

Since *policies* can be referenced by their identifiers, it is the responsibility of the *PAP* to ensure that these are unique. Confusion between identifiers could lead to misidentification of the *applicable policy*. This specification is silent on whether a *PAP* must generate a new identifier when a *policy* is modified or may use the same identifier in the modified *policy*. This is a matter of administrative practice. However, care must be taken in either case. If the identifier is reused, there is a danger that other *policies* or *policy sets* that reference it may be adversely affected. Conversely, if a new identifier is used, these other *policies* may continue to use the prior *policy*, unless it is deleted. In either case the results may not be what the *policy* administrator intends.

9.2.6. Trust model

Discussions of authentication, integrity and confidentiality safeguards necessarily assume an underlying trust model: how can one actor come to believe that a given key is uniquely associated with a specific, identified actor so that the key can be used to encrypt data for that actor or verify signatures (or other integrity structures) from that actor? Many different types of trust model exist, including strict hierarchies, distributed authorities, the Web, the bridge and so on.

It is worth considering the relationships between the various actors of the **access control** system in terms of the interdependencies that do and do not exist.

- None of the entities of the authorization system are dependent on the *PEP*. They may
 collect data from it, for example authentication data, but are responsible for verifying it
 themselves.
- The correct operation of the system depends on the ability of the *PEP* to actually enforce policy decisions.
- The PEP depends on the PDP to correctly evaluate policies. This in turn implies that the PDP is supplied with the correct inputs. Other than that, the PDP does not depend on the PEP.
- The **PDP** depends on the **PAP** to supply appropriate policies. The **PAP** is not dependent on other components.

9.2.7. Privacy

It is important to be aware that any transactions that occur with respect to **access control** may reveal private information about the actors. For example, if an XACML **policy** states that certain data may only be read by **subjects** with "Gold Card Member" status, then any transaction in which a **subject** is permitted **access** to that data leaks information to an adversary about the **subject's** status. Privacy considerations may therefore lead to encryption and/or to access control requirements surrounding the enforcement of XACML **policy** instances themselves: confidentiality-protected channels for the request/response protocol messages, protection of **subject attributes** in storage and in transit, and so on.

10. Conformance (normative)

10.1. Introduction

3863

3864

3865

3866

3867

3872

3873

3874 3875

3876

3877

3879

- 3868 The XACML specification addresses the following aspect of conformance:
- The XACML specification defines a number of functions, etc. that have somewhat special application, therefore they are not required to be implemented in an implementation that claims to conform with the OASIS standard.

10.2.Conformance tables

- This section lists those portions of the specification that MUST be included in an implementation of a *PDP* that claims to conform with XACML v2.0. A set of test cases has been created to assist in this process. These test cases are hosted by Sun Microsystems and can be located from the XACML Web page. The site hosting the test cases contains a full description of the test cases and how to execute them.
- Note: "M" means mandatory-to-implement. "O" means optional.

10.2.1. Schema elements

3880 The implementation MUST support those schema elements that are marked "M".

Element name	M/O
xacml-context:Action	M
xacml-context:Attribute	M
xacml-context:AttributeValue	M
xacml-context:Decision	M
xacml-context:Environment	M
xacml-context:MissingAttributeDetail	M
xacml-context:Obligations	0
xacml-context:Request	M
xacml-context:Resource	M
xacml-context:ResourceContent	0
xacml-context:Response	M
xacml-context:Result	M
xacml-context:Status	M
xacml-context:StatusCode	M
xacml-context:StatusDetail	0
xacml-context:StatusMessage	0
xacml-context:Subject	M
xacml:Action	M
xacml:ActionAttributeDesignator	M
xacml:ActionMatch	M
xacml:Actions	M
xacml:Apply	M
xacml:AttributeAssignment	0
xacml:AttributeSelector	0
xacml:AttributeValue	M

```
xacml:CombinerParameters
                                        0
xacml:CombinerParameter
                                        0
xacml:Condition
                                        Μ
xacml:Description
                                        M
xacml:Environment
                                        Μ
xacml:EnvironmentMatch
                                        M
xacml:EnvironmentAttributeDesignator
                                        M
xacml:Environments
                                        M
xacml:Expression
                                        Μ
xacml:Function
                                        М
xacml:Obligation
                                        0
xacml:Obligations
                                        0
xacml:Policy
                                        Μ
xacml:PolicyCombinerParameters
                                        0
xacml:PolicyDefaults
                                        0
xacml:PolicyIdReference
                                        Μ
xacml:PolicySet
                                        Μ
xacml:PolicySetDefaults
                                        0
xacml:PolicySetIdReference
                                        M
xacml:Resource
                                        Μ
xacml:ResourceAttributeDesignator
                                        Μ
xacml:ResourceMatch
                                        Μ
xacml:Resources
                                        M
xacml:Rule
                                        M
xacml:RuleCombinerParameters
                                        0
xacml:Subject
                                        Μ
xacml:SubjectMatch
                                        Μ
xacml:Subjects
                                        Μ
xacml:Target
                                        Μ
xacml:VariableDefinition
                                        Μ
xacml:VariableReference
                                        Μ
xacml:XPathVersion
                                        0
```

3881 **10.2.2. Identifier Prefixes**

3882 The following identifier prefixes are reserved by XACML.

```
Identifier
urn:oasis:names:tc:xacml:2.0
urn:oasis:names:tc:xacml:2.0:conformance-test
urn:oasis:names:tc:xacml:2.0:context
urn:oasis:names:tc:xacml:2.0:example
urn:oasis:names:tc:xacml:1.0:function
urn:oasis:names:tc:xacml:2.0:function
urn:oasis:names:tc:xacml:2.0:policy
urn:oasis:names:tc:xacml:1.0:subject
urn:oasis:names:tc:xacml:1.0:resource
urn:oasis:names:tc:xacml:1.0:action
urn:oasis:names:tc:xacml:1.0:environment
urn:oasis:names:tc:xacml:1.0:status
```

10.2.3. Algorithms

3883

3884

3885

The implementation MUST include the rule- and policy-combining algorithms associated with the following identifiers that are marked "M".

3889

3890

3891

3892

3893

3894

3895

```
Algorithm
                                                                          M/O
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides
                                                                           M
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides
                                                                           M
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-
                                                                           M
urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable
                                                                           M
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-
                                                                           M
applicable
urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-
                                                                           M
overrides
urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-
                                                                           M
overrides
urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-
                                                                           Μ
overrides
urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-
                                                                           Μ
overrides
```

3887 **10.2.4. Status Codes**

Implementation support for the <StatusCode> element is optional, but if the element is supported, then the following status codes must be supported and must be used in the way XACML has specified.

Identifier	M/O
urn:oasis:names:tc:xacml:1.0:status:missing-attribute	M
urn:oasis:names:tc:xacml:1.0:status:ok	M
urn:oasis:names:tc:xacml:1.0:status:processing-error	M
urn:oasis:names:tc:xacml:1.0:status:syntax-error	M

10.2.5. Attributes

The implementation MUST support the *attributes* associated with the following identifiers as specified by XACML. If values for these *attributes* are not present in the *decision request*, then their values MUST be supplied by the *context handler*. So, unlike most other *attributes*, their semantics are not transparent to the *PDP*.

Identifier	M/O
urn:oasis:names:tc:xacml:1.0:environment:current-time	M
urn:oasis:names:tc:xacml:1.0:environment:current-date	M
urn:oasis:names:tc:xacml:1.0:environment:current-dateTime	M

10.2.6. Identifiers

The implementation MUST use the *attributes* associated with the following identifiers in the way XACML has defined. This requirement pertains primarily to implementations of a *PAP* or *PEP* that uses XACML, since the semantics of the attributes are transparent to the *PDP*.

3899 3900

3896

3897

3898

3901

```
Identifier
                                                                          M/O
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name
                                                                           0
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address
                                                                           0
urn:oasis:names:tc:xacml:1.0:subject:authentication-method
                                                                           0
urn:oasis:names:tc:xacml:1.0:subject:authentication-time
                                                                           0
urn:oasis:names:tc:xacml:1.0:subject:key-info
                                                                           0
urn:oasis:names:tc:xacml:1.0:subject:request-time
                                                                           0
urn:oasis:names:tc:xacml:1.0:subject:session-start-time
                                                                           0
urn:oasis:names:tc:xacml:1.0:subject:subject-id
                                                                           0
urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier
                                                                           \cap
urn:oasis:names:tc:xacml:1.0:subject-category:access-subject
                                                                           M
urn:oasis:names:tc:xacml:1.0:subject-category:codebase
                                                                           0
urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject
                                                                           0
urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject
                                                                           0
urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine
                                                                           0
urn:oasis:names:tc:xacml:1.0:resource:resource-location
                                                                           0
urn:oasis:names:tc:xacml:1.0:resource:resource-id
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:resource:simple-file-name
                                                                           0
urn:oasis:names:tc:xacml:1.0:action:action-id
                                                                           0
urn:oasis:names:tc:xacml:1.0:action:implied-action
                                                                           0
```

3904 **10.2.7. Data-types**

The implementation MUST support the data-types associated with the following identifiers marked "M".

```
Data-type
                                                                          M/O
http://www.w3.org/2001/XMLSchema#string
                                                                           M
http://www.w3.org/2001/XMLSchema#boolean
                                                                           M
http://www.w3.org/2001/XMLSchema#integer
                                                                           M
http://www.w3.org/2001/XMLSchema#double
                                                                           M
http://www.w3.org/2001/XMLSchema#time
                                                                           Μ
http://www.w3.org/2001/XMLSchema#date
                                                                           M
http://www.w3.org/2001/XMLSchema#dateTime
                                                                           M
http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration
                                                                           M
http://www.w3.org/TR/2002/WD-xquery-operators-
                                                                           Μ
20020816#yearMonthDuration
http://www.w3.org/2001/XMLSchema#anyURI
                                                                           Μ
http://www.w3.org/2001/XMLSchema#hexBinary
                                                                           Μ
http://www.w3.org/2001/XMLSchema#base64Binary
                                                                           M
urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:data-type:x500Name
                                                                           Μ
```

10.2.8. Functions

The implementation MUST properly process those functions associated with the identifiers marked with an "M".

```
FunctionM/Ourn:oasis:names:tc:xacml:1.0:function:string-equalMurn:oasis:names:tc:xacml:1.0:function:boolean-equalMurn:oasis:names:tc:xacml:1.0:function:integer-equalMurn:oasis:names:tc:xacml:1.0:function:double-equalMurn:oasis:names:tc:xacml:1.0:function:date-equalMurn:oasis:names:tc:xacml:1.0:function:time-equalM
```

```
urn:oasis:names:tc:xacml:1.0:function:dateTime-equal
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-equal
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:anyURI-equal
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:x500Name-equal
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:hexBinary-equal
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-equal
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:integer-add
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:double-add
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:integer-subtract
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:double-subtract
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:integer-multiply
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:double-multiply
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:integer-divide
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:double-divide
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:integer-mod
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:integer-abs
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:double-abs
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:round
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:floor
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:string-normalize-space
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:double-to-integer
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:integer-to-double
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:or
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:and
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:n-of
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:not
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:integer-greater-than
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:integer-less-than
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:double-greater-than
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:double-less-than
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-
                                                                           M
yearMonthDuration
urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:string-greater-than
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:string-less-than
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:time-greater-than
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:time-less-than
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal
                                                                           M
urn:oasis:names:tc:xacml:2.0:function:time-in-range
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than
                                                                           M
```

```
urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal
urn:oasis:names:tc:xacml:1.0:function:date-greater-than
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:date-less-than
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:string-one-and-only
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:string-bag-size
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:string-is-in
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:string-bag
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:boolean-one-and-only
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:boolean-bag-size
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:boolean-is-in
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:boolean-bag
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:integer-bag-size
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:integer-is-in
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:integer-bag
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:double-one-and-only
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:double-baq-size
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:double-is-in
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:double-bag
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:time-one-and-only
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:time-bag-size
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:time-is-in
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:time-bag
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:date-one-and-only
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:date-bag-size
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:date-is-in
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:date-bag
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:dateTime-one-and-only
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:dateTime-bag-size
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:dateTime-is-in
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:dateTime-bag
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:anyURI-one-and-only
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:anyURI-bag-size
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:anyURI-is-in
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:anyURI-bag
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-one-and-only
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-bag-size
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:hexBinary-is-in
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-bag
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-one-and-only
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-bag-size
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:base64Binary-is-in
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:base64Binary-bag
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-one-and-only
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-bag-size
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-is-in
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-bag
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-one-and-only
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-bag-size
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-is-in
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-bag
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:x500Name-one-and-only
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:x500Name-bag-size
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:x500Name-is-in
                                                                           M
```

```
urn:oasis:names:tc:xacml:1.0:function:x500Name-bag
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-one-and-only
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-bag-size
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-is-in
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-baq
                                                                           M
urn:oasis:names:tc:xacml:2.0:function:string-concatenate
                                                                           Μ
urn:oasis:names:tc:xacml:2.0:function:uri-string-concatenate
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:any-of
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:all-of
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:any-of-any
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:all-of-any
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:any-of-all
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:all-of-all
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:map
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:x500Name-match
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:string-regexp-match
                                                                           M
urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match
                                                                           M
urn:oasis:names:tc:xacml:2.0:function:ipAddress-regexp-match
                                                                           Μ
urn:oasis:names:tc:xacml:2.0:function:dnsName-regexp-match
                                                                           M
urn:oasis:names:tc:xacml:2.0:function:rfc822Name-regexp-match
                                                                           M
urn:oasis:names:tc:xacml:2.0:function:x500Name-regexp-match
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:xpath-node-count
                                                                           0
urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal
                                                                           \cap
urn:oasis:names:tc:xacml:1.0:function:xpath-node-match
                                                                           \cap
urn:oasis:names:tc:xacml:1.0:function:string-intersection
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:string-union
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:string-subset
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:string-set-equals
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:boolean-intersection
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:boolean-at-least-one-member-of
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:boolean-union
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:boolean-subset
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:boolean-set-equals
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:integer-intersection
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:integer-at-least-one-member-of
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:integer-union
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:integer-subset
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:integer-set-equals
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:double-intersection
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:double-at-least-one-member-of
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:double-union
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:double-subset
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:double-set-equals
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:time-intersection
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:time-at-least-one-member-of
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:time-union
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:time-subset
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:time-set-equals
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:date-intersection
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:date-at-least-one-member-of
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:date-union
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:date-subset
                                                                           Μ
urn:oasis:names:tc:xacml:1.0:function:date-set-equals
                                                                           M
urn:oasis:names:tc:xacml:1.0:function:dateTime-intersection
                                                                           M
```

```
urn:oasis:names:tc:xacml:1.0:function:dateTime-at-least-one-member-of
urn:oasis:names:tc:xacml:1.0:function:dateTime-union
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:dateTime-subset
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:dateTime-set-equals
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:anyURI-intersection
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:anyURI-at-least-one-member-of
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:anyURI-union
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:anyURI-subset
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:anyURI-set-equals
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-intersection
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-at-least-one-member-of
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-union
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-subset
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:hexBinary-set-equals
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-intersection
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:base64Binary-at-least-one-member-
of
urn:oasis:names:tc:xacml:1.0:function:base64Binary-union
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-subset
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:base64Binary-set-equals
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-intersection
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-at-least-one-
                                                                          Μ
member-of
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-union
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-subset
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-set-equals
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-intersection
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-at-least-one-
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-union
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-subset
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-set-equals
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:x500Name-intersection
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:x500Name-at-least-one-member-of
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:x500Name-union
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:x500Name-subset
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:x500Name-set-equals
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-intersection
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-at-least-one-member-of
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-union
                                                                          Μ
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-subset
                                                                          M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-set-equals
                                                                          Μ
```

3911

11. References

3912 3913	[DS]	D. Eastlake et al., XML-Signature Syntax and Processing, http://www.w3.org/TR/xmldsig-core/ , World Wide Web Consortium.
3914 3915 3916	[Hancock]	Hancock, "Polymorphic Type Checking", in Simon L. Peyton Jones, "Implementation of Functional Programming Languages", Section 8, Prentice-Hall International, 1987
3917 3918	[Haskell]	Haskell, a purely functional language. Available at http://www.haskell.org/

3919 3920 3921	[Hier]	Anderson, A., ed., "Hierarchical resource profile of XACML v2.0", OASIS Standard, 1 February 2005, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-hier-profile-spec-os.pdf
3922 3923 3924	[Hinton94]	Hinton, H, M, Lee,, E, S, The Compatibility of Policies, Proceedings 2nd ACM Conference on Computer and Communications Security, Nov 1994, Fairfax, Virginia, USA.
3925 3926	[IEEE754]	IEEE Standard for Binary Floating-Point Arithmetic 1985, ISBN 1-5593-7653-8, IEEE Product No. SH10116-TBR
3927 3928 3929	[ISO10181-3]	ISO/IEC 10181-3:1996 Information technology – Open Systems Interconnection Security frameworks for open systems: Access control framework.
3930 3931 3932	[Kudo00]	Kudo M and Hada S, XML document security based on provisional authorization, Proceedings of the Seventh ACM Conference on Computer and Communications Security, Nov 2000, Athens, Greece, pp 87-96.
3933 3934	[LDAP-1]	RFC2256, A summary of the X500(96) User Schema for use with LDAPv3, Section 5, M Wahl, December 1997 http://www.ietf.org/rfc/rfc2798.txt
3935 3936	[LDAP-2]	RFC2798, Definition of the inetOrgPerson, M. Smith, April 2000 http://www.ietf.org/rfc/rfc2798.txt
3937 3938 3939	[MathML]	Mathematical Markup Language (MathML), Version 2.0, W3C Recommendation, 21 February 2001. Available at: http://www.w3.org/TR/MathML2/
3940 3941 3942	[Multi]	Anderson, A., ed., "Multiple resource profile of XACML v2.0", OASIS Standard, 1 February 2005, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-mult-profile-spec-os.pdf
3943 3944 3945 3946	[Perritt93]	Perritt, H. Knowbots, Permissions Headers and Contract Law, Conference on Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment, April 1993. Available at: http://www.ifla.org/documents/infopol/copyright/perh2.txt
3947 3948 3949	[RBAC]	Role-Based Access Controls, David Ferraiolo and Richard Kuhn, 15th National Computer Security Conference, 1992. Available at: http://csrc.nist.gov/rbac
3950 3951	[RegEx]	XML Schema Part 0: Primer, W3C Recommendation, 2 May 2001, Appendix D. Available at: http://www.w3.org/TR/xmlschema-0/
3952 3953	[RFC2119]	S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997
3954 3955	[RFC2396]	Berners-Lee T, Fielding R, Masinter L, Uniform Resource Identifiers (URI): Generic Syntax. Available at: http://www.ietf.org/rfc/rfc2396.txt
3956 3957	[RFC2732	Hinden R, Carpenter B, Masinter L, Format for Literal IPv6 Addresses in URL's. Available at: http://www.ietf.org/rfc/rfc2732.txt
3958 3959	[RFC3198]	IETF RFC 3198: Terminology for Policy-Based Management, November 2001. http://www.ietf.org/rfc/rfc3198.txt
3960 3961	[SAML]	Security Assertion Markup Language available from http://www.oasis- open.org/committees/security/#documents
3962 3963 3964	[Sloman94]	Sloman, M. Policy Driven Management for Distributed Systems. Journal of Network and Systems Management, Volume 2, part 4. Plenum Press. 1994.
3965 3966	[XACMLv1.0]	Extensible access control markup language (XACML) Version 1.0. OASIS Standard. 18 February 2003. Available at: http://www.oasis-

3967 3968		open.org/apps/org/workgroup/xacml/download.php/940/oasis-xacml-1.0.pdf
3969 3970 3971 3972	[XACMLv1.1]	Extensible access control markup language (XACML) Version 1.1. OASIS Committee Specification. 7 August 2003. Available at: http://www.oasis-open.org/apps/org/workgroup/xacml/download.php/4104/cs-xacml-specification-1.1.pdf
3973 3974 3975	[XF]	XQuery 1.0 and XPath 2.0 Functions and Operators, W3C Working Draft 16 August 2002. Available at: http://www.w3.org/TR/2002/WD-xquery-operators-20020816
3976 3977 3978	[XS]	XML Schema, parts 1 and 2. Available at: http://www.w3.org/TR/xmlschema-1/ and http://www.w3.org/TR/xmlschema-2/
3979 3980	[XPath]	XML Path Language (XPath), Version 1.0, W3C Recommendation 16 November 1999. Available at: http://www.w3.org/TR/xpath
3981 3982 3983	[XSLT]	XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999. Available at: http://www.w3.org/TR/xslt

3984 Appendix A. Data-types and functions (normative)

3985	A.1. Introduction		
3986 3987	This section specifies the data-types and functions used in XACML to create <i>predicates</i> for <i>conditions</i> and <i>target</i> matches.		
3988 3989 3990 3991	This specification combines the various standards set forth by IEEE and ANSI for string representation of numeric values, as well as the evaluation of arithmetic functions. It describes the primitive data-types and <i>bags</i> . The standard functions are named and their operational semantics are described.		
3992	A.2. Data-types		
3993 3994 3995 3996 3997	Although XML instances represent all data-types as strings, an XACML <i>PDP</i> must reason about types of data that, while they have string representations, are not just strings. Types such as Boolean, integer and double MUST be converted from their XML string representations to values that can be compared with values in their domain of discourse, such as numbers. The following primitive data-types are specified for use with XACML and have explicit data representations:		
3998	 http://www.w3.org/2001/XMLSchema#string 		
3999	 http://www.w3.org/2001/XMLSchema#boolean 		
4000	 http://www.w3.org/2001/XMLSchema#integer 		
4001	 http://www.w3.org/2001/XMLSchema#double 		
4002	 http://www.w3.org/2001/XMLSchema#time 		
4003	 http://www.w3.org/2001/XMLSchema#date 		
4004	 http://www.w3.org/2001/XMLSchema#dateTime 		
4005	 http://www.w3.org/2001/XMLSchema#anyURI 		
4006	 http://www.w3.org/2001/XMLSchema#hexBinary 		
4007	 http://www.w3.org/2001/XMLSchema#base64Binary 		
4008	 http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration 		
4009	 http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration 		
4010	urn:oasis:names:tc:xacml:1.0:data-type:x500Name		
4011	urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name		
4012	 urn:oasis:names:tc:xacml:2.0:data-type:ipAddress 		
4013	urn:oasis:names:tc:xacml:2.0:data-type:dnsName		
4014 4015	For the sake of improved interoperability, it is RECOMMENDED that all time references be in UTC time.		

access_control-xacml-2.0-core-spec-os Copyright © OASIS Open 2004. All Rights Reserved.

- 4016 An XACML PDP SHALL be capable of converting string representations into various primitive data-
- 4017 types. For integers and doubles, XACML SHALL use the conversions described in [IEEE754].
- 4018 XACML defines three data-types; these are:
- 4019 "urn:oasis:names:tc:xacml:1.0:data-type:x500Name",
- 4020 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"
- 4021 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress"
- 4022 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" and
- 4023 These types represent identifiers for subjects or resources and appear in several standard
- 4024 applications, such as TLS/SSL and electronic mail.
- 4025 X.500 directory name
- 4026 The "urn:oasis:names:tc:xacml:1.0:data-type:x500Name" primitive type represents an ITU-T Rec.
- 4027 X.520 Distinguished Name. The valid syntax for such a name is described in IETF RFC 2253
- 4028 "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names"
- 4029 RFC 822 name
- 4030 The "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" primitive type represents an electronic
- 4031 mail address. The valid syntax for such a name is described in IETF RFC 2821, Section 4.1.2,
- 4032 Command Argument Syntax, under the term "Mailbox".
- 4033 IP address
- The "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" primitive type represents an IPv4 or IPv6 network address, with optional mask and optional port or port range. The syntax SHALL be:
- 4036 4037 ipAddress = address ["/" mask] [":" [portrange]]

- For an IPv4 address, the address and mask are formatted in accordance with the syntax for a
- 4040 "host" in IETF RFC 2396 "Uniform Resource Identifiers (URI): Generic Syntax", section 3.2.
- 4041 For an IPv6 address, the address and mask are formatted in accordance with the syntax for an
- "ipv6reference" in IETF RFC 2732 "Format for Literal IPv6 Addresses in URL's". (Note that an IPv6 address or mask, in this syntax, is enclosed in literal "[" "]" brackets.)
- 4044

4046

4047

4048

4045 **DNS name**

The "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" primitive type represents a Domain Name Service (DNS) host name, with optional port or port range. The syntax SHALL be:

```
dnsName = hostname [ ":" portrange ]
```

4049 4050 4051

4052

The hostname is formatted in accordance with IETF RFC 2396 "Uniform Resource Identifiers (URI): Generic Syntax", section 3.2, except that a wildcard "*" may be used in the left-most component of the hostname to indicate "any subdomain" under the domain specified to its right.

4053 4054 4055

- For both the "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" and
- 4056 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" data-types, the port or port range syntax 4057 SHALL be

4058

4059 portrange = portnumber | "-"portnumber | portnumber"-"[portnumber]

4060

where "portnumber" is a decimal port number. If the port number is of the form "-x", where "x" is a port number, then the range is all ports numbered "x" and below. If the port number is of the form

4063 4064	"x-", then the range is all ports numbered "x" and above. [This syntax is taken from the Java SocketPermission.]
4065	A.3. Functions
4066 4067	XACML specifies the following functions. If an argument of one of these functions were to evaluate to "Indeterminate", then the function SHALL be set to "Indeterminate".
4068	A.3.1 Equality predicates
4069 4070	The following functions are the <i>equality</i> functions for the various primitive types. Each function for a particular data-type follows a specified standard convention for that data-type.
4071	urn:oasis:names:tc:xacml:1.0:function:string-equal
4072 4073 4074 4075 4076 4077	This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if the value of both of its arguments are of equal length and each string is determined to be equal byte-by-byte according to the function "integer-equal". Otherwise, it SHALL return "False".
4078	urn:oasis:names:tc:xacml:1.0:function:boolean-equal
4079 4080 4081 4082	This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#boolean" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if the arguments are equal. Otherwise, it SHALL return "False".
4083	urn:oasis:names:tc:xacml:1.0:function:integer-equal
4084 4085 4086 4087	This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#integer" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation on integers according to IEEE 754 [IEEE 754].
4088	urn:oasis:names:tc:xacml:1.0:function:double-equal
4089 4090 4091 4092	This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#double" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation on doubles according to IEEE 754 [IEEE 754].
4093	urn:oasis:names:tc:xacml:1.0:function:date-equal
4094 4095 4096 4097	This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#date" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to the "op:date-equal" function [XF Section 8.3.11].
4098	urn:oasis:names:tc:xacml:1.0:function:time-equal
4099 4100 4101 4102	This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to the "op:time-equal" function [XF Section 8.3.14].

4103	•	urn:oasis:names:tc:xacml:1.0:function:dateTime-equal
4104 4105 4106 4107		This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to the "op:dateTime-equal" function [XF Section 8.3.8].
4108	•	urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal
4109 4110 4111 4112 4113 4114		This function SHALL take two arguments of data-type "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation according to the "op:dayTimeDuration-equal" function [XF Section 8.3.5]. Note that the lexical representation of each argument MUST be converted to a value expressed in fractional seconds [XF Section 8.2.2].
4115	•	urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-equal
4116 4117 4118 4119 4120 4121		This function SHALL take two arguments of data-type "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation according to the "op:yearMonthDuration-equal" function [XF Section 8.3.2]. Note that the lexical representation of each argument MUST be converted to a value expressed in integer months [XF Section 8.2.1].
4122	•	urn:oasis:names:tc:xacml:1.0:function:anyURI-equal
4123 4124 4125 4126		This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to the "op:anyURI-equal" function [XF Section 10.2.1].
4127	•	urn:oasis:names:tc:xacml:1.0:function:x500Name-equal
4128 4129 4130 4131 4132		This function SHALL take two arguments of "urn:oasis:names:tc:xacml:1.0:data-type:x500Name" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if each Relative Distinguished Name (RDN) in the two arguments matches. Otherwise, it SHALL return "False". Two RDNs shall be said to match if and only if the result of the following operations is "True" ³ .
4133 4134		 Normalize the two arguments according to IETF RFC 2253 "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names".
4135 4136 4137		 If any RDN contains multiple attributeTypeAndValue pairs, re-order the Attribute ValuePairs in that RDN in ascending order when compared as octet strings (described in ITU-T Rec. X.690 (1997 E) Section 11.6 "Set-of components").
4138 4139 4140		 Compare RDNs using the rules in IETF RFC 3280 "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", Section 4.1.2.4 "Issuer".
4141	•	urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal
4142 4143		This function SHALL take two arguments of data-type "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean".

access_control-xacml-2.0-core-spec-os

³ ITU-T Rec. X.520 contains rules for matching X500 names, but these are very complex and require knowledge of the syntax of various AttributeTypes. IETF RFC 3280 contains simplified matching rules that the XACML x500Name-equal function uses.

4144 4145 4146 4147	It SHALL return "True" if and only if the two arguments are equal. Otherwise, it SHALL return "False". An RFC822 name consists of a <i>local-part</i> followed by "@" followed by a <i>domain-part</i> . The <i>local-part</i> is case-sensitive, while the <i>domain-part</i> (which is usually a DNS host name) is not case-sensitive. Perform the following operations:
4148	1. Normalize the domain-part of each argument to lower case
4149 4150	 Compare the expressions by applying the function "urn:oasis:names:tc:xacml:1.0:function:string-equal" to the normalized arguments.
4151	urn:oasis:names:tc:xacml:1.0:function:hexBinary-equal
4152 4153 4154 4155 4156 4157 4158 4159	This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#hexBinary" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet sequences represented by the value of both arguments have equal length and are equal in a conjunctive, point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function. Otherwise, it SHALL return "False". The conversion from the string representation to an octet sequence SHALL be as specified in [XS Section 8.2.15].
4160	 urn:oasis:names:tc:xacml:1.0:function:base64Binary-equal
4161 4162 4163 4164 4165 4166 4167 4168	This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#base64Binary" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet sequences represented by the value of both arguments have equal length and are equal ir a conjunctive, point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function. Otherwise, it SHALL return "False". The conversion from the string representation to an octet sequence SHALL be as specified in [XS Section 8.2.16].
4169	A.3.2 Arithmetic functions
4170 4171 4172 4173 4174 4175 4176	All of the following functions SHALL take two arguments of the specified <i>data-type</i> , integer or double, and SHALL return an element of integer or double data-type, respectively. However, the "add" functions MAY take more than two arguments. Each function evaluation SHALL proceed as specified by their logical counterparts in IEEE 754 [IEEE 754]. In an expression that contains any of these functions, if any argument is "Indeterminate", then the expression SHALL evaluate to "Indeterminate". In the case of the divide functions, if the divisor is zero, then the function SHALL evaluate to "Indeterminate".
4177	urn:oasis:names:tc:xacml:1.0:function:integer-add
4178	This function MAY have two or more arguments.
4179	urn:oasis:names:tc:xacml:1.0:function:double-add
4180	This function MAY have two or more arguments.
4181	urn:oasis:names:tc:xacml:1.0:function:integer-subtract
4182	urn:oasis:names:tc:xacml:1.0:function:double-subtract
4183	urn:oasis:names:tc:xacml:1.0:function:integer-multiply
4184	urn:oasis:names:tc:xacml:1.0:function:double-multiply
4185	urn:oasis:names:tc:xacml:1.0:function:integer-divide

4186	urn:oasis:names:tc:xacml:1.0:function:double-divide
4187	urn:oasis:names:tc:xacml:1.0:function:integer-mod
4188 4189 4190 4191	The following functions SHALL take a single argument of the specified <i>data-type</i> . The round and floor functions SHALL take a single argument of data-type "http://www.w3.org/2001/XMLSchema#double" and return a value of the data-type "http://www.w3.org/2001/XMLSchema#double".
4192	urn:oasis:names:tc:xacml:1.0:function:integer-abs
4193	urn:oasis:names:tc:xacml:1.0:function:double-abs
4194	urn:oasis:names:tc:xacml:1.0:function:round
4195	urn:oasis:names:tc:xacml:1.0:function:floor
4196	A.3.3 String conversion functions
4197 4198	The following functions convert between values of the data-type "http://www.w3.org/2001/XMLSchema#string" primitive types.
4199	urn:oasis:names:tc:xacml:1.0:function:string-normalize-space
4200 4201 4202	This function SHALL take one argument of data-type "http://www.w3.org/2001/XMLSchema#string" and SHALL normalize the value by stripping off all leading and trailing white space characters.
4203	urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case
4204 4205 4206	This function SHALL take one argument of data-type "http://www.w3.org/2001/XMLSchema#string" and SHALL normalize the value by converting each upper case character to its lower case equivalent.
4207	A.3.4 Numeric data-type conversion functions
4208 4209 4210	The following functions convert between the data-type "http://www.w3.org/2001/XMLSchema#integer" and" http://www.w3.org/2001/XMLSchema#double" primitive types.
4211	urn:oasis:names:tc:xacml:1.0:function:double-to-integer
4212 4213 4214 4215	This function SHALL take one argument of data-type "http://www.w3.org/2001/XMLSchema#double" and SHALL truncate its numeric value to a whole number and return an element of data-type "http://www.w3.org/2001/XMLSchema#integer".
4216	urn:oasis:names:tc:xacml:1.0:function:integer-to-double
4217 4218 4219	This function SHALL take one argument of data-type "http://www.w3.org/2001/XMLSchema#integer" and SHALL promote its value to an element of data-type "http://www.w3.org/2001/XMLSchema#double" with the same numeric value.
4220	A.3.5 Logical functions
4221	This section contains the specification for logical functions that operate on arguments of data-type

"http://www.w3.org/2001/XMLSchema#boolean".

- urn:oasis:names:tc:xacml:1.0:function:or
- This function SHALL return "False" if it has no arguments and SHALL return "True" if at least one of its arguments evaluates to "True". The order of evaluation SHALL be from first argument to last. The evaluation SHALL stop with a result of "True" if any argument evaluates to "True", leaving the rest of the arguments unevaluated.
- 4228 urn:oasis:names:tc:xacml:1.0:function:and
- This function SHALL return "True" if it has no arguments and SHALL return "False" if one of its arguments evaluates to "False". The order of evaluation SHALL be from first argument to last. The evaluation SHALL stop with a result of "False" if any argument evaluates to "False", leaving the rest of the arguments unevaluated.
- urn:oasis:names:tc:xacml:1.0:function:n-of
- 4234 The first argument to this function SHALL be of data-type 4235 http://www.w3.org/2001/XMLSchema#integer. The remaining arguments SHALL be of 4236 data-type http://www.w3.org/2001/XMLSchema#boolean. The first argument specifies the minimum number of the remaining arguments that MUST evaluate to "True" for the 4237 expression to be considered "True". If the first argument is 0, the result SHALL be "True". 4238 4239 If the number of arguments after the first one is less than the value of the first argument, 4240 then the expression SHALL result in "Indeterminate". The order of evaluation SHALL be: 4241 first evaluate the integer value, then evaluate each subsequent argument. The evaluation 4242 SHALL stop and return "True" if the specified number of arguments evaluate to "True". The 4243 evaluation of arguments SHALL stop if it is determined that evaluating the remaining 4244 arguments will not satisfy the requirement.
- 4245 urn:oasis:names:tc:xacml:1.0:function:not

4256

- This function SHALL take one argument of data-type

 "http://www.w3.org/2001/XMLSchema#boolean". If the argument evaluates to "True", then
 the result of the expression SHALL be "False". If the argument evaluates to "False", then
 the result of the expression SHALL be "True".
- Note: When evaluating and, or, or n-of, it MAY NOT be necessary to attempt a full evaluation of each argument in order to determine whether the evaluation of the argument would result in "Indeterminate". Analysis of the argument regarding the availability of its attributes, or other analysis regarding errors, such as "divide-by-zero", may render the argument error free. Such arguments occurring in the expression in a position after the evaluation is stated to stop need not be processed.

A.3.6 Numeric comparison functions

- These functions form a minimal set for comparing two numbers, yielding a Boolean result. They SHALL comply with the rules governed by IEEE 754 [IEEE 754].
- urn:oasis:names:tc:xacml:1.0:function:integer-greater-than
- urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal
- urn:oasis:names:tc:xacml:1.0:function:integer-less-than
- urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal
- urn:oasis:names:tc:xacml:1.0:function:double-greater-than
- urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal

• urn:oasis:names:tc:xacml:1.0:function:double-less-than

4267

• urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal

A.3.7 Date and time arithmetic functions

- These functions perform arithmetic operations with date and time.
- urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration
- This function SHALL take two arguments, the first SHALL be of data-type

 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be of data-type

 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration". It SHALL

 return a result of "http://www.w3.org/2001/XMLSchema#dateTime". This function SHALL

 return the value by adding the second argument to the first argument according to the

 specification of adding durations to date and time [XS Appendix E].
- urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration
- This function SHALL take two arguments, the first SHALL be a

 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a

 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration". It

 SHALL return a result of "http://www.w3.org/2001/XMLSchema#dateTime". This function

 SHALL return the value by adding the second argument to the first argument according to

 the specification of adding durations to date and time [XS Appendix E].
- urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration
- 4284 This function SHALL take two arguments, the first SHALL be a 4285 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a 4286 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration". It SHALL 4287 return a result of "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument 4288 is a positive duration, then this function SHALL return the value by adding the 4289 corresponding negative duration, as per the specification [XS Appendix E]. If the second argument is a negative duration, then the result SHALL be as if the function 4290 4291 "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration" had been applied 4292 to the corresponding positive duration.
- urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-yearMonthDuration
- 4294 This function SHALL take two arguments, the first SHALL be a 4295 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a 4296 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration". It 4297 SHALL return a result of "http://www.w3.org/2001/XMLSchema#dateTime". If the second 4298 argument is a positive duration, then this function SHALL return the value by adding the 4299 corresponding negative duration, as per the specification [XS Appendix E]. If the second argument is a negative duration, then the result SHALL be as if the function 4300 4301 "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration" had been 4302 applied to the corresponding positive duration.
- urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration
- This function SHALL take two arguments, the first SHALL be a

 "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a

 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration". It

 SHALL return a result of "http://www.w3.org/2001/XMLSchema#date". This function

 SHALL return the value by adding the second argument to the first argument according to

 the specification of adding duration to date [XS Appendix E].

4310	•	urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration
4311 4312 4313 4314 4315 4316 4317 4318 4319		This function SHALL take two arguments, the first SHALL be a "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration". It SHALL return a result of "http://www.w3.org/2001/XMLSchema#date". If the second argument is a positive duration, then this function SHALL return the value by adding the corresponding negative duration, as per the specification [XS Appendix E]. If the second argument is a negative duration, then the result SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration" had been applied to the corresponding positive duration.
4320		A.3.8 Non-numeric comparison functions
4321	Th	ese functions perform comparison operations on two arguments of non-numerical types.
4322	•	urn:oasis:names:tc:xacml:1.0:function:string-greater-than
4323 4324 4325 4326 4327 4328 4329 4330 4331		This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the arguments are compared byte by byte and, after an initial prefix of corresponding bytes from both arguments that are considered equal by "urn:oasis:names:tc:xacml:1.0:function:integer-equal", the next byte by byte comparison is such that the byte from the first argument is greater than the byte from the second argument by the use of the function "urn:oasis:names:tc:xacml:2.0:function:integer-greater then". Otherwise, it SHALL return "False".
4332	•	urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal
4333 4334 4335 4336 4337 4338		This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return a result as if evaluated with the logical function "urn:oasis:names:tc:xacml:1.0:function:or" with two arguments containing the functions "urn:oasis:names:tc:xacml:1.0:function:string-greater-than" and "urn:oasis:names:tc:xacml:1.0:function:string-equal" containing the original arguments
4339	•	urn:oasis:names:tc:xacml:1.0:function:string-less-than
4340 4341 4342 4343 4344 4345 4346 4347 4348		This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the arguments are compared byte by byte and, after an initial prefix of corresponding bytes from both arguments that are considered equal by "urn:oasis:names:tc:xacml:1.0:function:integer-equal", the next byte by byte comparison is such that the byte from the first argument is less than the byte from the second argument by the use of the function "urn:oasis:names:tc:xacml:1.0:function:integer-less-than". Otherwise, it SHALL return "False".
4349	•	urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal
4350 4351 4352 4353 4354 4355		This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return a result as if evaluated with the function "urn:oasis:names:tc:xacml:1.0:function:or" with two arguments containing the functions "urn:oasis:names:tc:xacml:1.0:function:string-less-than" and "urn:oasis:names:tc:xacml:1.0:function:string-equal" containing the original arguments.

4356	•	urn:oasis:names:tc:xacml:1.0:function:time-greater-than
4357 4358 4359 4360 4361 4362 4363		This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first argument is greater than the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time" [XS Section 3.2.8]. Otherwise, it SHALL return "False". Note: it is illegal to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.
4364	•	urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal
4365 4366 4367 4368 4369 4370 4371 4372		This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first argument is greater than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time" [XS Section 3.2.8]. Otherwise, it SHALL return "False". Note: it is illegal to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.
4373	•	urn:oasis:names:tc:xacml:1.0:function:time-less-than
4374 4375 4376 4377 4378 4379 4380		This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first argument is less than the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time" [XS Section 3.2.8]. Otherwise, it SHALL return "False". Note: it is illegal to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.
4381	•	urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal
4382 4383 4384 4385 4386 4387 4388		This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first argument is less than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#time" [XS Section 3.2.8]. Otherwise, it SHALL return "False". Note: it is illegal to compare a time that includes a time-zone value with one that does not. In such cases, the time-in-range function should be used.
4389	•	urn:oasis:names:tc:xacml:1.0:function:time-in-range
4390 4391 4392 4393 4394 4395 4396 4397 4398		This function SHALL take three arguments of data-type "http://www.w3.org/2001/XMLSchema#time" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the first argument falls in the range defined inclusively by the second and third arguments. Otherwise, it SHALL return "False". Regardless of its value, the third argument SHALL be interpreted as a time that is equal to, or later than by less than twenty-four hours, the second argument. If no time zone is provided for the first argument, it SHALL use the default time zone at the context handler. If no time zone is provided for the second or third arguments, then they SHALL use the time zone from the first argument.
4399	•	urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than
4400 4401 4402		This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the

4403 first argument is greater than the second argument according to the order relation specified 4404 for "http://www.w3.org/2001/XMLSchema#dateTime" by [XF Section 3.2.7]. Otherwise, it 4405 SHALL return "False". Note: if a dateTime value does not include a time-zone value, then 4406 an implicit time-zone value SHALL be assigned, as described in [XF]. 4407 urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal 4408 This function SHALL take two arguments of data-type 4409 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an 4410 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the 4411 first argument is greater than or equal to the second argument according to the order 4412 relation specified for "http://www.w3.org/2001/XMLSchema#dateTime" by [XF Section 3.2.7]. Otherwise, it SHALL return "False". Note: if a dateTime value does not include a 4413 4414 time-zone value, then an implicit time-zone value SHALL be assigned, as described in 4415 [XF]. 4416 urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than 4417 This function SHALL take two arguments of data-type 4418 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the 4419 4420 first argument is less than the second argument according to the order relation specified for 4421 "http://www.w3.org/2001/XMLSchema#dateTime" by [XF Section 3.2.7]. Otherwise, it 4422 SHALL return "False". Note: if a dateTime value does not include a time-zone value, then 4423 an implicit time-zone value SHALL be assigned, as described in [XF]. 4424 urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal 4425 This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema# dateTime" and SHALL return an 4426 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the 4427 4428 first argument is less than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#dateTime" by [XF Section 3.2.7]. 4429 4430 Otherwise, it SHALL return "False". Note: if a dateTime value does not include a time-zone 4431 value, then an implicit time-zone value SHALL be assigned, as described in [XF]. 4432 urn:oasis:names:tc:xacml:1.0:function:date-greater-than 4433 This function SHALL take two arguments of data-type 4434 "http://www.w3.org/2001/XMLSchema#date" and SHALL return an 4435 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the 4436 first argument is greater than the second argument according to the order relation specified 4437 for "http://www.w3.org/2001/XMLSchema#date" by [XF Section 3.2.9]. Otherwise, it SHALL 4438 return "False". Note: if a date value does not include a time-zone value, then an implicit 4439 time-zone value SHALL be assigned, as described in [XF]. 4440 urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal 4441 This function SHALL take two arguments of data-type 4442 "http://www.w3.org/2001/XMLSchema#date" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the 4443 4444 first argument is greater than or equal to the second argument according to the order 4445 relation specified for "http://www.w3.org/2001/XMLSchema#date" by [XF Section 3.2.9]. Otherwise, it SHALL return "False". Note: if a date value does not include a time-zone 4446 4447 value, then an implicit time-zone value SHALL be assigned, as described in [XF].

urn:oasis:names:tc:xacml:1.0:function:date-less-than

4449 4450 4451 4452 4453 4454 4455	This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#date" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first argument is less than the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#date" by [XF Section 3.2.9]. Otherwise, it SHALL return "False". Note: if a date value does not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described in [XF].
4456	urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal
4457 4458 4459 4460 4461 4462 4463	This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#date" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first argument is less than or equal to the second argument according to the order relation specified for "http://www.w3.org/2001/XMLSchema#date" by [XF Section 3.2.9]. Otherwise it SHALL return "False". Note: if a date value does not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described in [XF].
4464	A.3.9 String functions
4465	The following functions operate on strings and URIs.
4466 4467	urn:oasis:names:tc:xacml:2.0:function:string-concatenate
4468 4469 4470 4471	This function SHALL take two or more arguments of data-type "http://www.w3.org/2001/XMLSchema#string" and SHALL return a "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the concatenation, in order, of the arguments.
4472	urn:oasis:names:tc:xacml:2.0:function:url-string-concatenate
4473 4474 4475 4476 4477	This function SHALL take one argument of data-type "http://www.w3.org/2001/XMLSchema#anyURI" and one or more arguments of type "http://www.w3.org/2001/XMLSchema#string", and SHALL return a "http://www.w3.org/2001/XMLSchema#anyURI". The result SHALL be the URI constructed by appending, in order, the "string" arguments to the "anyURI" argument.
4478	A.3.10 Bag functions
4479 4480 4481	These functions operate on a bag of 'type' values, where type is one of the primitive data-types. Some additional conditions defined for each function below SHALL cause the expression to evaluate to "Indeterminate".
4482	urn:oasis:names:tc:xacml:1.0:function: <i>type</i> -one-and-only
4483 4484 4485	This function SHALL take a bag of 'type' values as an argument and SHALL return a value of '-type'. It SHALL return the only value in the bag . If the bag does not have one and only one value, then the expression SHALL evaluate to "Indeterminate".
4486	urn:oasis:names:tc:xacml:1.0:function: <i>type-</i> bag-size
4487 4488	This function SHALL take a bag of 'type' values as an argument and SHALL return an "http://www.w3.org/2001/XMLSchema#integer" indicating the number of values in the bag .
4489	urn:oasis:names:tc:xacml:1.0:function: <i>type</i> -is-in

4490 This function SHALL take an argument of 'type' as the first argument and a bag of type 4491 values as the second argument and SHALL return an 4492 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL evaluate to "True" if 4493 and only if the first argument matches by the "urn:oasis:names:tc:xacml:x.x:function:type-4494 equal" any value in the **baq**. Otherwise, it SHALL return "False". 4495 urn:oasis:names:tc:xacml:1.0:function:type-bag 4496 This function SHALL take any number of arguments of 'type' and return a **bag** of 'type' 4497 values containing the values of the arguments. An application of this function to zero 4498 arguments SHALL produce an empty bag of the specified data-type. A.3.11 Set functions 4499 4500 These functions operate on bags mimicking sets by eliminating duplicate elements from a bag. 4501 urn:oasis:names:tc:xacml:1.0:function:type-intersection 4502 This function SHALL take two arguments that are both a bag of 'type' values. It SHALL 4503 return a bag of 'type' values such that it contains only elements that are common between 4504 the two bags, which is determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal". No duplicates, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal", 4505 SHALL exist in the result. 4506 4507 urn:oasis:names:tc:xacml:1.0:function:type-at-least-one-member-of 4508 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL evaluate to 4509 4510 "True" if and only if at least one element of the first argument is contained in the second 4511 argument as determined by "urn:oasis:names:tc:xacml:x.x:function:type-is-in". 4512 urn:oasis:names:tc:xacml:1.0:function:type-union 4513 This function SHALL take two arguments that are both a **bag** of 'type' values. The 4514 expression SHALL return a bag of 'type' such that it contains all elements of both bags. No duplicates, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal", 4515 4516 SHALL exist in the result. 4517 urn:oasis:names:tc:xacml:1.0:function:type-subset 4518 This function SHALL take two arguments that are both a bag of 'type' values. It SHALL 4519 return a "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and 4520 only if the first argument is a subset of the second argument. Each argument SHALL be 4521 considered to have had its duplicates removed, as determined by 4522 "urn:oasis:names:tc:xacml:x.x:function:type-equal", before the subset calculation. 4523 urn:oasis:names:tc:xacml:1.0:function:type-set-equals 4524 This function SHALL take two arguments that are both a bag of 'type' values. It SHALL return a "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return the result of 4525 applying "urn:oasis:names:tc:xacml:1.0:function:and" to the application of 4526 "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the first and second arguments and 4527 4528 the application of "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the second and

first arguments.

A.3.12 Higher-order bag functions

- This section describes functions in XACML that perform operations on *bags* such that functions may be applied to the *bags* in general.
- In this section, a general-purpose functional language called Haskell [Haskell] is used to formally specify the semantics of these functions. Although the English description is adequate, a formal
- 4535 specification of the semantics is helpful.
- 4536 For a quick summary, in the following Haskell notation, a function definition takes the form of
- 4537 clauses that are applied to patterns of structures, namely lists. The symbol "[]" denotes the empty
- 4538 list, whereas the expression "(x:xs)" matches against an argument of a non-empty list of which "x"
- represents the first element of the list, and "xs" is the rest of the list, which may be an empty list.
- We use the Haskell notion of a list, which is an ordered collection of elements, to model the XACML
- 4541 **bags** of values.

4530

4559

4560

4561

4562

4563

4564

4565 4566

4567

4568

4569

- 4542 A simple Haskell definition of a familiar function "urn:oasis:names:tc:xacml:1.0:function:and" that
- takes a list of values of type Boolean is defined as follows:
- 4544 and:: [Bool] -> Bool
- 4545 and [] = True
- 4546 and (x:xs) = x && (and xs)
- The first definition line denoted by a "::" formally describes the data-type of the function, which takes a list of Booleans, denoted by "[Bool]", and returns a Boolean, denoted by "Bool". The second
- definition line is a clause that states that the function "and" applied to the empty list is "True". The
- 4550 third definition line is a clause that states that for a non-empty list, such that the first element is "x", 4551 which is a value of data-type Bool, the function "and" applied to x SHALL be combined with, using
- 4551 which is a value of data-type Bool, the function "and" applied to x SHALL be combined with, using the logical conjunction function, which is denoted by the infix symbol "&&", the result of recursively
- 4553 applying the function "and" to the rest of the list. Of course, an application of the "and" function is
- True" if and only if the list to which it is applied is empty or every element of the list is "True". For
- example, the evaluation of the following Haskell expressions,
- 4556 (and []), (and [True]), (and [True,True]), (and [True,True,False])
- 4557 evaluate to "True", "True", "True", and "False", respectively.
- 4558 urn:oasis:names:tc:xacml:1.0:function:any-of

This function applies a Boolean function between a specific primitive value and a **bag** of values, and SHALL return "True" if and only if the predicate is "True" for at least one element of the **bag**.

In Haskell, the semantics of this operation are as follows:

```
4570 any_of :: (a -> b -> Bool) -> a -> [b] -> Bool

4571 any_of f a [] = False

4572 any_of f a (x:xs) = (f a x) || (any_of f a xs)
```

In the above notation, "f" is the function to be applied, "a" is the primitive value, and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression SHALL return "True":

```
4576
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
4577
          <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
4578
          <AttributeValue
4579
       DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>
4580
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
4581
             <AttributeValue
4582
       DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>
4583
             <AttributeValue
4584
       DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>
4585
             <AttributeValue
4586
       DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>
4587
             <AttributeValue
4588
       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>
4589
          </Apply>
4590
       </Apply>
```

This expression is "True" because the first argument is equal to at least one of the elements of the *bag*, according to the function.

urn:oasis:names:tc:xacml:1.0:function:all-of

4575

4591

4592

4593

4594

4595 4596

4597

4598 4599

4600 4601

4602 4603

4607

4608 4609 This function applies a Boolean function between a specific primitive value and a **bag** of values, and returns "True" if and only if the predicate is "True" for every element of the **bag**.

In Haskell, the semantics of this operation are as follows:

```
4604 all\_of :: (a -> b -> Bool) -> a -> [b] -> Bool
4605 all\_of f a [] = True
4606 all\_of f a (x:xs) = (f a x) && (all\_of f a xs)
```

In the above notation, "f" is the function to be applied, "a" is the primitive value, and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression SHALL evaluate to "True":

```
4610
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of">
4611
          <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-greater"/>
4612
          <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
4613
4614
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4615
             <AttributeValue
4616
       DataType="http://www.w3.org/2001/XMLSchema#integer">9</AttributeValue>
4617
             <AttributeValue
4618
       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4619
             <AttributeValue
4620
       DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
4621
             <AttributeValue
4622
       DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
4623
          </Apply>
4624
       </Apply>
```

This expression is "True" because the first argument (10) is greater than *all* of the elements of the *bag* (9,3,4 and 2).

urn:oasis:names:tc:xacml:1.0:function:any-of-any

4627

4628

4629 4630

4631

4632

4633

4634

4635

4636

4637

4638

4639 4640

4641

4642

4643

4644

4645

4646 4647

4667

4668 4669

4670

4671

4672

This function applies a Boolean function between each element of a **bag** of values and each element of another **bag** of values, and returns "True" if and only if the predicate is "True" for at least one comparison.

In Haskell, taking advantage of the "any_of" function defined above, the semantics of the "any_of_any" function are as follows:

```
any_of_any :: ( a -> b -> Bool ) -> [a]-> [b] -> Bool
any_of_any f [] ys = False
any_of_any f (x:xs) ys = (any_of f x ys) || (any_of_any f xs ys)
```

In the above notation, "f" is the function to be applied and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression SHALL evaluate to "True":

```
4648
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-any">
4649
          <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
4650
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
4651
             <AttributeValue
4652
       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>
4653
             <AttributeValue
4654
       DataType="http://www.w3.org/2001/XMLSchema#string">Mary</AttributeValue>
4655
          </Apply>
4656
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
4657
            <AttributeValue
4658
       DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>
4659
            <AttributeValue
4660
       DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>
4661
            <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>
4662
4663
             <AttributeValue
4664
       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>
4665
          </Apply>
4666
       </Apply>
```

This expression is "True" because at least one of the elements of the first *bag*, namely "Ringo", is equal to at least one of the elements of the second *bag*.

urn:oasis:names:tc:xacml:1.0:function:all-of-any

This function applies a Boolean function between the elements of two *bags*. The expression SHALL be "True" if and only if the supplied predicate is 'True' between each element of the first *bag* and any element of the second *bag*.

This function SHALL take three arguments. The first argument SHALL be an <xacml:Function> element that names a Boolean function that takes two arguments of primitive types. The second argument SHALL be a *bag* of a primitive data-type. The third argument SHALL be a *bag* of a primitive data-type. The expression SHALL be evaluated as if the "urn:oasis:names:tc:xacml:1.0:function:any-of" function had been applied to each value of the first *bag* and the whole of the second *bag* using the supplied xacml:Function, and the results were then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

In Haskell, taking advantage of the "any_of" function defined in Haskell above, the semantics of the "all_of_any" function are as follows:

```
all_of_any :: ( a -> b -> Bool ) -> [a]-> [b] -> Bool
all_of_any f [] ys = True
all of any f (x:xs) ys = (any of f x ys) && (all of any f xs ys)
```

In the above notation, "f" is the function to be applied and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression SHALL evaluate to "True":

```
4688
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-any">
4689
          <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-greater"/>
4690
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4691
             <AttributeValue
4692
       DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
4693
             <AttributeValue
4694
       DataType="http://www.w3.org/2001/XMLSchema#integer">20</AttributeValue>
4695
          </Apply>
4696
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4697
             <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4698
4699
             <AttributeValue
4700
       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4701
             <AttributeValue
4702
       DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4703
             <AttributeValue
4704
       DataType="http://www.w3.org/2001/XMLSchema#integer">19</AttributeValue>
4705
          </Apply>
4706
       </Apply>
```

This expression is "True" because each of the elements of the first *bag* is greater than at least one of the elements of the second *bag*.

urn:oasis:names:tc:xacml:1.0:function:any-of-all

4673

4674

4675

4676

4677 4678

4679 4680

4681 4682

4683

4684

4685

4686

4687

4707

4708

4709

4710

4711

4712

4713

4714

4715

4716 4717

4718

4719

4720 4721 This function applies a Boolean function between the elements of two *bags*. The expression SHALL be "True" if and only if the supplied predicate is "True" between each element of the second *bag* and any element of the first *bag*.

In Haskell, taking advantage of the "all_of" function defined in Haskell above, the semantics of the "any_of_all" function are as follows:

```
4722 any_of_all :: (a \rightarrow b \rightarrow Bool) \rightarrow [a] \rightarrow [b] \rightarrow Bool
4723 any_of_all f [] ys = False
4724 any_of_all f (x:xs)_ys = (all_of_f x_ys)_|| (any_of_all_f xs_ys)_||
```

In the above notation, "f" is the function name to be applied and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression SHALL evaluate to "True":

```
4728
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-all">
4729
          <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-greater"/>
4730
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4731
             <AttributeValue
4732
       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4733
             <AttributeValue
4734
       DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4735
          4736
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4737
             <AttributeValue
4738
       DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4739
             <AttributeValue
4740
       DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
4741
            <AttributeValue
4742
       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4743
             <AttributeValue
4744
       DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
4745
          </Apply>
4746
       </Apply>
```

This expression is "True" because, for all of the values in the second *bag*, there is a value in the first *bag* that is greater.

urn:oasis:names:tc:xacml:1.0:function:all-of-all

4725

4726 4727

4747

4748

4749

4750

4751

4752

4753

4754

4755

4756

4757

4758

4759

4760 4761

4762

4763

4767

4768 4769 This function applies a Boolean function between the elements of two *bags*. The expression SHALL be "True" if and only if the supplied predicate is "True" between each and every element of the first *bag* collectively against all the elements of the second *bag*.

In Haskell, taking advantage of the "all_of" function defined in Haskell above, the semantics of the "all_of_all" function is as follows:

In the above notation, "f" is the function to be applied and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression SHALL evaluate to "True":

```
4770
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-all">
4771
          <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-greater"/>
4772
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4773
             <AttributeValue
4774
       DataType="http://www.w3.org/2001/XMLSchema#integer">6</AttributeValue>
4775
             <AttributeValue
4776
       DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4777
          </larable>
4778
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4779
             <AttributeValue
4780
       DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4781
             <AttributeValue
4782
       DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
4783
             <AttributeValue
4784
       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4785
            <AttributeValue
4786
       DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
4787
          </Apply>
4788
       </Apply>
```

This expression is "True" because all elements of the first *bag*, "5" and "6", are each greater than all of the integer values "1", "2", "3", "4" of the second *bag*.

urn:oasis:names:tc:xacml:1.0:function:map

4789

4790 4791

4792

4793

4794

4795

4796 4797

4798

4799

4800

4804 4805

4806

4817

This function converts a **bag** of values to another **bag** of values.

In Haskell, this function is defined as follows:

```
4801 map:: (a -> b) -> [a] -> [b]
4802 map f [] = []
4803 map f (x:xs) = (f x): (map f xs)
```

In the above notation, "f" is the function to be applied and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression,

```
4807
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:map">
4808
          <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-normalize-</pre>
4809
       to-lower-case">
4810
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
4811
             <AttributeValue
4812
       DataType="http://www.w3.org/2001/XMLSchema#string">Hello</AttributeValue>
4813
             <AttributeValue
4814
       DataType="http://www.w3.org/2001/XMLSchema#string">World!</AttributeValue>
4815
          </Apply>
4816
       </Apply>
```

evaluates to a **bag** containing "hello" and "world!".

4818 A.3.13 Regular-expression-based functions

- These functions operate on various types using regular expressions and evaluate to 4820 "http://www.w3.org/2001/XMLSchema#boolean".
- 4821 urn:oasis:names:tc:xacml:1.0:function:string-regexp-match
- This function decides a regular expression match. It SHALL take two arguments of
 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
 expression and the second argument SHALL be a general string. The function
 specification SHALL be that of the "xf:matches" function with the arguments reversed [XF]
- specification SHALL be that of the "xf:matches" function with the arguments reversed [XF Section 6.3.15].
- urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match
- This function decides a regular expression match. It SHALL take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "http://www.w3.org/2001/XMLSchema#anyURI". It SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular expression and the second argument SHALL be a URI. The function SHALL convert the second argument to type "http://www.w3.org/2001/XMLSchema#string", then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- urn:oasis:names:tc:xacml:2.0:function:ipAddress-regexp-match
- This function decides a regular expression match. It SHALL take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". It SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular expression and the second argument SHALL be an IPv4 or IPv6 address. The function SHALL convert the second argument to type "http://www.w3.org/2001/XMLSchema#string", then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- urn:oasis:names:tc:xacml:2.0:function:dnsName-regexp-match
- This function decides a regular expression match. It SHALL take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". It SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular expression and the second argument SHALL be a DNS name. The function SHALL convert the second argument to type "http://www.w3.org/2001/XMLSchema#string", then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
 - urn:oasis:names:tc:xacml:2.0:function:rfc822Name-regexp-match
- This function decides a regular expression match. It SHALL take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". It SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular expression and the second argument SHALL be an RFC 822 name. The function SHALL convert the second argument to type "http://www.w3.org/2001/XMLSchema#string", then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- urn:oasis:names:tc:xacml:2.0:function:x500Name-regexp-match
- This function decides a regular expression match. It SHALL take two arguments; the first is of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type

4863 4864 4865 4866 4867	"urn:oasis:names:tc:xacml:1.0:data-type:x500Name". It SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular expression and the second argument SHALL be an X.500 directory name. The function SHALL convert the second argument to type "http://www.w3.org/2001/XMLSchema#string", then apply "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
4868	A.3.14 Special match functions
4869 4870 4871	These functions operate on various types and evaluate to "http://www.w3.org/2001/XMLSchema#boolean" based on the specified standard matching algorithm.
4872	urn:oasis:names:tc:xacml:1.0:function:x500Name-match
4873 4874 4875 4876	This function shall take two arguments of "urn:oasis:names:tc:xacml:2.0:data-type:x500Name" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and only if the first argument matches some terminal sequence of RDNs from the second argument when compared using x500Name-equal.
4877	urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match
4878 4879 4880 4881 4882	This function SHALL take two arguments, the first is of data-type "http://www.w3.org/2001/XMLSchema#string" and the second is of data-type "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". This function SHALL evaluate to "True" if the first argument matches the second argument according to the following specification.
4883 4884 4885	An RFC822 name consists of a local-part followed by "@" followed by a domain-part. The local-part is case-sensitive, while the domain-part (which is usually a DNS name) is not case-sensitive. ⁴
4886 4887	The second argument contains a complete rfc822Name. The first argument is a complete or partial rfc822Name used to select appropriate values in the second argument as follows.
4888 4889 4890 4891 4892	In order to match a particular address in the second argument, the first argument must specify the complete mail address to be matched. For example, if the first argument is "Anderson@sun.com", this matches a value in the second argument of "Anderson@sun.com" and "Anderson@SUN.COM", but not "Anne.Anderson@sun.com", "anderson@sun.com" or "Anderson@east.sun.com".
4893 4894 4895 4896	In order to match any address at a particular domain in the second argument, the first argument must specify only a domain name (usually a DNS name). For example, if the first argument is "sun.com", this matches a value in the first argument of "Anderson@sun.com" or "Baxter@SUN.COM", but not "Anderson@east.sun.com".
4897 4898 4899 4900 4901	In order to match any address in a particular domain in the second argument, the first argument must specify the desired domain-part with a leading ".". For example, if the first argument is ".east.sun.com", this matches a value in the second argument of "Anderson@east.sun.com" and "anne.anderson@ISRG.EAST.SUN.COM" but not "Anderson@sun.com".

access_control-xacml-2.0-core-spec-os

⁴ According to IETF RFC822 and its successor specifications [RFC2821], case is significant in the *local-part*. Many mail systems, as well as the IETF PKIX specification, treat the *local-part* as case-insensitive. This anomaly is considered an error by mail-system designers and is not encouraged. For this reason, rfc822Name-match treats *local-part* as case sensitive.

4902 A.3.15 XPath-based functions

urn:oasis:names:tc:xacml:1.0:function:xpath-node-count

This function SHALL take an "http://www.w3.org/2001/XMLSchema#string" as an argument, which SHALL be interpreted as an XPath expression, and evaluates to an "http://www.w3.org/2001/XMLSchema#integer". The value returned from the function SHALL be the count of the nodes within the node-set that match the given XPath expression.

urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal

This function SHALL take two "http://www.w3.org/2001/XMLSchema#string" arguments, which SHALL be interpreted as XPath expressions, and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if any of the XML nodes in the node-set matched by the first argument equals, according to the "op:node-equal" function [XF Section 13.1.6], any of the XML nodes in the node-set matched by the second argument.

urn:oasis:names:tc:xacml:1.0:function:xpath-node-match

This function SHALL take two "http://www.w3.org/2001/XMLSchema#string" arguments, which SHALL be interpreted as XPath expressions and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". This function SHALL evaluate to "True" if one of the following two conditions is satisfied: (1) Any of the XML nodes in the node-set matched by the first argument is equal, according to "op:node-equal" [XF Section 13.1.6], to any of the XML nodes in the node-set matched by the second argument; (2) any attribute and element node below any of the XML nodes in the node-set matched by the first argument is equal, according to "op:node-equal" [XF Section 13.1.6], to any of the XML nodes in the node-set matched by the second argument.

NOTE: The first condition is equivalent to "xpath-node-equal", and guarantees that "xpath-node-equal" is a special case of "xpath-node-match".

A.3.16 Extension functions and primitive types

- Functions and primitive types are specified by string identifiers allowing for the introduction of functions in addition to those specified by XACML. This approach allows one to extend the XACML module with special functions and special primitive data-types.
- 4938 In order to preserve the integrity of the XACML evaluation strategy, the result of an extension
- 4939 function SHALL depend only on the values of its arguments. Global and hidden parameters SHALL
- 4940 NOT affect the evaluation of an expression. Functions SHALL NOT have side effects, as
- 4941 evaluation order cannot be guaranteed in a standard way.

4942 Appendix B. XACML identifiers (normative)

4943 This section defines standard identifiers for commonly used entities.

B.1. XACML namespaces

- 4945 There are currently two defined XACML namespaces.
- 4946 Policies are defined using this identifier.

4944

4950

4968

- 4947 urn:oasis:names:tc:xacml:2.0:policy:schema:os
- 4948 Request and response *contexts* are defined using this identifier.
- 4949 urn:oasis:names:tc:xacml:2.0:context:schema:os

B.2. Access subject categories

- This identifier indicates the system entity that initiated the *access* request. That is, the initial entity
- 4952 in a request chain. If **subject** category is not specified, this is the default value.
- 4953 urn:oasis:names:tc:xacml:1.0:subject-category:access-subject
- This identifier indicates the system entity that will receive the results of the request (used when it is
- 4955 distinct from the access-subject).
- 4956 urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject
- 4957 This identifier indicates a system entity through which the *access* request was passed. There may
- 4958 be more than one. No means is provided to specify the order in which they passed the message.
- 4959 urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject
- 4960 This identifier indicates a system entity associated with a local or remote codebase that generated
- 4961 the request. Corresponding subject attributes might include the URL from which it was loaded
- 4962 and/or the identity of the code-signer. There may be more than one. No means is provided to
- 4963 specify the order in which they processed the request.
- 4964 urn:oasis:names:tc:xacml:1.0:subject-category:codebase
- 4965 This identifier indicates a system entity associated with the computer that initiated the access
- 4966 request. An example would be an IPsec identity.
- 4967 urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine

B.3. Data-types

- 4969 The following identifiers indicate data-types that are defined in Section A.2.
- 4970 urn:oasis:names:tc:xacml:1.0:data-type:x500Name.
- 4971 urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name
- 4972 urn:oasis:names:tc:xacml:2.0:data-type:ipAddress
- 4973 urn:oasis:names:tc:xacml:2.0:data-type:dnsName
- The following data-type identifiers are defined by XML Schema [XS].
- 4975 http://www.w3.org/2001/XMLSchema#string
- 4976 http://www.w3.org/2001/XMLSchema#boolean

```
4977
       http://www.w3.org/2001/XMLSchema#integer
4978
       http://www.w3.org/2001/XMLSchema#double
4979
       http://www.w3.org/2001/XMLSchema#time
4980
       http://www.w3.org/2001/XMLSchema#date
4981
       http://www.w3.org/2001/XMLSchema#dateTime
4982
       http://www.w3.org/2001/XMLSchema#anyURI
4983
       http://www.w3.org/2001/XMLSchema#hexBinary
4984
       http://www.w3.org/2001/XMLSchema#base64Binary
4985
       The following data-type identifiers correspond to the dayTimeDuration and yearMonthDuration
4986
       data-types defined in [XF Sections 8.2.2 and 8.2.1, respectively].
4987
       http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration
4988
       http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration
```

B.4. Subject attributes

- These identifiers indicate *attributes* of a *subject*. When used, they SHALL appear within a
- 4991 <Subject> element of the request *context*. They SHALL be accessed by means of a
- 4992 <SubjectAttributeDesignator> element, or an <AttributeSelector> element that points
- 4993 into a <Subject> element of the request *context*.
- 4994 At most one of each of these attributes is associated with each subject. Each attribute associated
- 4995 with authentication included within a single <Subject> element relates to the same authentication
- 4996 event.

- 4997 This identifier indicates the name of the *subject*. The default format is
- 4998 "http://www.w3.org/2001/XMLSchema#string". To indicate other formats, use the DataType
- 4999 attributes listed in B.3
- 5000 urn:oasis:names:tc:xacml:1.0:subject:subject-id
- 5001 This identifier indicates the **subject** category. "access-subject" is the default value.
- 5002 urn:oasis:names:tc:xacml:1.0:subject-category
- 5003 This identifier indicates the security domain of the *subject*. It identifies the administrator and policy
- that manages the name-space in which the *subject* id is administered.
- 5005 urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier
- 5006 This identifier indicates a public key used to confirm the *subject's* identity.
- 5007 urn:oasis:names:tc:xacml:1.0:subject:key-info
- This identifier indicates the time at which the *subject* was authenticated.
- 5009 urn:oasis:names:tc:xacml:1.0:subject:authentication-time
- 5010 This identifier indicates the method used to authenticate the **subject**.
- 5011 urn:oasis:names:tc:xacml:1.0:subject:authn-locality:authentication-method
- This identifier indicates the time at which the **subject** initiated the **access** request, according to the
- 5013 **PEP**.
- 5014 urn:oasis:names:tc:xacml:1.0:subject:request-time
- 5015 This identifier indicates the time at which the **subject's** current session began, according to the
- 5016 **PEP**.
- 5017 urn:oasis:names:tc:xacml:1.0:subject:session-start-time
- 5018 The following identifiers indicate the location where authentication credentials were activated. They
- 5019 are intended to support the corresponding entities from the SAML authentication statement
- 5020 **[SAML]**.

5021	This identifier indicates that the location is expressed as an IP address.
5022	urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address
5023	The corresponding attribute SHALL be of data-type "http://www.w3.org/2001/XMLSchema#string".
5024	This identifier indicates that the location is expressed as a DNS name.
5025	urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name
5026	The corresponding attribute SHALL be of data-type "http://www.w3.org/2001/XMLSchema#string".
5027 5028 5029	Where a suitable attribute is already defined in LDAP [LDAP-1, LDAP-2], the XACML identifier SHALL be formed by adding the <i>attribute</i> name to the URI of the LDAP specification. For example, the <i>attribute</i> name for the userPassword defined in the RFC 2256 SHALL be:
5030	http://www.ietf.org/rfc/rfc2256.txt#userPassword
5031	B.6. Resource attributes
5032 5033 5034	These identifiers indicate <i>attributes</i> of the <i>resource</i> . The corresponding <i>attributes</i> MAY appear in the <resource> element of the request <i>context</i> and be accessed by means of a <resourceattributedesignator> element, or by an <attributeselector> element that</attributeselector></resourceattributedesignator></resource>
5035	points into the <resource> element of the request <i>context</i>.</resource>
5036	This attribute identifies the resource to which access is requested. If an <xacml-< td=""></xacml-<>
5037 5038	context:ResourceContent> element is provided, then the resource to which access is requested SHALL be all or a portion of the resource supplied in the <xacml-< td=""></xacml-<>
5039	context: ResourceContent> element.
5040	urn:oasis:names:tc:xacml:1.0:resource:resource-id
5041 5042 5043 5044 5045	This attribute identifies the namespace of the top element of the contents of the <xacml-context:resourcecontent> element. In the case where the resource content is supplied in the request context and the resource namespace is defined in the resource, the PDP SHALL confirm that the namespace defined by this attribute is the same as that defined in the resource. The type of the corresponding attribute SHALL be "http://www.w3.org/2001/XMLSchema#anyURI".</xacml-context:resourcecontent>
5046	urn:oasis:names:tc:xacml:2.0:resource:target-namespace
5047	B.7. Action attributes
5048 5049 5050 5051	These identifiers indicate attributes of the action being requested. When used, they SHALL appear within the <action> element of the request context. They SHALL be accessed by means of an <actionattributedesignator> element, or an <attributeselector> element that points into the <action> element of the request context.</action></attributeselector></actionattributedesignator></action>
5052	This attribute identifies the action for which access is requested.
5053	urn:oasis:names:tc:xacml:1.0:action:action-id
5054	Where the action is implicit, the value of the action-id attribute SHALL be
5055	urn:oasis:names:tc:xacml:1.0:action:implied-action
5056	This attribute identifies the namespace in which the agt i on it attribute is defined

urn:oasis:names:tc:xacml:1.0:action:action-namespace

B.8. Environment attributes 5058 5059 These identifiers indicate *attributes* of the *environment* within which the *decision request* is to be 5060 evaluated. When used in the decision request, they SHALL appear in the <Environment> 5061 element of the request *context*. They SHALL be accessed by means of an 5062 <EnvironmentAttributeDesignator> element, or an <AttributeSelector> element that 5063 points into the <Environment> element of the request context. 5064 This identifier indicates the current time at the *context handler*. In practice it is the time at which 5065 the request *context* was created. For this reason, if these identifiers appear in multiple places 5066 within a <Policy> or <PolicySet>, then the same value SHALL be assigned to each occurrence 5067 in the evaluation procedure, regardless of how much time elapses between the processing of the occurrences. 5068 5069 urn:oasis:names:tc:xacml:1.0:environment:current-time 5070 The corresponding *attribute* SHALL be of data-type 5071 "http://www.w3.org/2001/XMLSchema#time". 5072 urn:oasis:names:tc:xacml:1.0:environment:current-date 5073 The corresponding attribute SHALL be of data-type 5074 "http://www.w3.org/2001/XMLSchema#date". 5075 urn:oasis:names:tc:xacml:1.0:environment:current-dateTime 5076 The corresponding attribute SHALL be of data-type 5077 "http://www.w3.org/2001/XMLSchema#dateTime". B.9. Status codes 5078 5079 The following status code values are defined. 5080 This identifier indicates success. 5081 urn:oasis:names:tc:xacml:1.0:status:ok 5082 This identifier indicates that all the attributes necessary to make a policy decision were not available 5083 (see Section 6.16). 5084 urn:oasis:names:tc:xacml:1.0:status:missing-attribute 5085 This identifier indicates that some attribute value contained a syntax error, such as a letter in a numeric field. 5086 5087 urn:oasis:names:tc:xacml:1.0:status:syntax-error 5088 This identifier indicates that an error occurred during policy evaluation. An example would be 5089 division by zero. 5090 urn:oasis:names:tc:xacml:1.0:status:processing-error

B.10.Combining algorithms

- The deny-overrides rule-combining algorithm has the following value for the
- 5093 ruleCombiningAlgId attribute:

5091

5094 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides

- 5095 The deny-overrides policy-combining algorithm has the following value for the
- 5096 policyCombiningAlgId attribute:
- 5097 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides
- 5098 The permit-overrides rule-combining algorithm has the following value for the
- 5099 ruleCombiningAlgId attribute:
- 5100 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides
- 5101 The permit-overrides policy-combining algorithm has the following value for the
- 5102 policyCombiningAlgId attribute:
- 5103 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides
- 5104 The first-applicable rule-combining algorithm has the following value for the
- 5105 ruleCombiningAlgId attribute:
- 5106 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable
- 5107 The first-applicable policy-combining algorithm has the following value for the
- 5108 policyCombiningAlgId attribute:
- 5109 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable
- 5110 The only-one-applicable-policy policy-combining algorithm has the following value for the
- 5111 policyCombiningAlgId attribute:
- 5112 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-
- 5113 applicable
- 5114 The ordered-deny-overrides rule-combining algorithm has the following value for the
- 5115 ruleCombiningAlgId attribute:
- 5116 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-
- 5117 overrides
- 5118 The ordered-deny-overrides policy-combining algorithm has the following value for the
- 5119 policyCombiningAlgId attribute:
- 5120 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-
- 5121 overrides
- 5122 The ordered-permit-overrides rule-combining algorithm has the following value for the
- 5123 ruleCombiningAlgId attribute:
- 5124 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-
- 5125 overrides
- 5126 The ordered-permit-overrides policy-combining algorithm has the following value for the
- 5127 policyCombiningAlgId attribute:
- 5128 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-
- 5129 overrides

Appendix C. Combining algorithms (normative)

5131 This section contains a description of the *rule*- and *policy-combining algorithms* specified by 5132 XACML.

C.1. Deny-overrides

5130

5133

5134 5135

5136 5137

5138

5139

5140

5141

5142

5143

5144

5145 5146

5147

5148

The following specification defines the "Deny-overrides" *rule-combining algorithm* of a *policy*.

In the entire set of *rules* in the *policy*, if any *rule* evaluates to "Deny", then the result of the *rule* combination SHALL be "Deny". If any *rule* evaluates to "Permit" and all other *rules* evaluate to "NotApplicable", then the result of the *rule* combination SHALL be "Permit". In other words, "Deny" takes precedence, regardless of the result of evaluating any of the other *rules* in the combination. If all *rules* are found to be "NotApplicable" to the *decision request*, then the *rule* combination SHALL evaluate to "NotApplicable".

If an error occurs while evaluating the *target* or *condition* of a *rule* that contains an *effect* value of "Deny" then the evaluation SHALL continue to evaluate subsequent *rules*, looking for a result of "Deny". If no other *rule* evaluates to "Deny", then the combination SHALL evaluate to "Indeterminate", with the appropriate error status.

If at least one *rule* evaluates to "Permit", all other *rules* that do not have evaluation errors evaluate to "Permit" or "NotApplicable" and all *rules* that do have evaluation errors contain *effects* of "Permit", then the result of the combination SHALL be "Permit".

The following pseudo-code represents the evaluation strategy of this *rule-combining algorithm*.

```
5149
       Decision denyOverridesRuleCombiningAlgorithm(Rule rule[])
5150
5151
          Boolean atLeastOneError = false;
5152
          Boolean potentialDeny
                                    = false;
          Boolean atLeastOnePermit = false;
5153
5154
          for( i=0 ; i < lengthOf(rules) ; i++ )</pre>
5155
5156
             Decision decision = evaluate(rule[i]);
5157
             if (decision == Deny)
5158
5159
                return Deny;
5160
5161
             if (decision == Permit)
5162
5163
                atLeastOnePermit = true;
5164
                continue;
5165
5166
             if (decision == NotApplicable)
5167
             {
5168
                continue;
5169
5170
             if (decision == Indeterminate)
5171
5172
                atLeastOneError = true;
5173
5174
                if (effect(rule[i]) == Deny)
5175
5176
                   potentialDeny = true;
5177
```

```
5178
                continue;
5179
             }
5180
5181
          if (potentialDeny)
5182
          {
5183
             return Indeterminate;
5184
5185
          if (atLeastOnePermit)
5186
          {
5187
             return Permit;
5188
5189
          if (atLeastOneError)
5190
          {
5191
             return Indeterminate;
5192
5193
          return NotApplicable;
5194
       }
```

The following specification defines the "Deny-overrides" *policy-combining algorithm* of a *policy set*

In the entire set of *policies* in the *policy set*, if any *policy* evaluates to "Deny", then the result of the *policy* combination SHALL be "Deny". In other words, "Deny" takes precedence, regardless of the result of evaluating any of the other *policies* in the *policy set*. If all *policies* are found to be "NotApplicable" to the *decision request*, then the *policy set* SHALL evaluate to "NotApplicable".

If an error occurs while evaluating the *target* of a *policy*, or a reference to a *policy* is considered invalid or the *policy* evaluation results in "Indeterminate", then the *policy set* SHALL evaluate to "Deny".

The following pseudo-code represents the evaluation strategy of this *policy-combining algorithm*.

```
5206
       Decision denyOverridesPolicyCombiningAlgorithm(Policy policy[])
5207
5208
          Boolean atLeastOnePermit = false;
5209
          for( i=0 ; i < lengthOf(policy) ; i++ )</pre>
5210
5211
             Decision decision = evaluate(policy[i]);
5212
             if (decision == Deny)
5213
5214
                return Deny;
5215
5216
             if (decision == Permit)
5217
             {
5218
                atLeastOnePermit = true;
5219
                continue;
5220
5221
             if (decision == NotApplicable)
5222
             {
5223
                continue;
5224
5225
             if (decision == Indeterminate)
5226
             {
5227
                return Deny;
5228
5229
5230
          if (atLeastOnePermit)
5231
5232
             return Permit;
5233
5234
          return NotApplicable;
```

519551965197

5198

5199

5200

5201

5202

5203

5204

Obligations of the individual **policies** shall be combined as described in Section 7.14.

C.2. Ordered-deny-overrides

The following specification defines the "Ordered-deny-overrides" *rule-combining algorithm* of a *policy*.

The behavior of this algorithm is identical to that of the Deny-overrides *rule-combining algorithm* with one exception. The order in which the collection of *rules* is evaluated SHALL match the order as listed in the *policy*.

The following specification defines the "Ordered-deny-overrides" *policy-combining algorithm* of a *policy set*.

The behavior of this algorithm is identical to that of the Deny-overrides **policy-combining algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL match the order as listed in the **policy set**.

C.3. Permit-overrides

The following specification defines the "Permit-overrides" *rule-combining algorithm* of a *policy*.

In the entire set of *rules* in the *policy*, if any *rule* evaluates to "Permit", then the result of the *rule* combination SHALL be "Permit". If any *rule* evaluates to "Deny" and all other *rules* evaluate to "NotApplicable", then the *policy* SHALL evaluate to "Deny". In other words, "Permit" takes precedence, regardless of the result of evaluating any of the other *rules* in the *policy*. If all *rules* are found to be "NotApplicable" to the *decision request*, then the *policy* SHALL evaluate to "NotApplicable".

If an error occurs while evaluating the *target* or *condition* of a *rule* that contains an *effect* of "Permit" then the evaluation SHALL continue looking for a result of "Permit". If no other *rule* evaluates to "Permit", then the *policy* SHALL evaluate to "Indeterminate", with the appropriate error status.

If at least one *rule* evaluates to "Deny", all other *rules* that do not have evaluation errors evaluate to "Deny" or "NotApplicable" and all *rules* that do have evaluation errors contain an *effect* value of "Deny", then the *policy* SHALL evaluate to "Deny".

The following pseudo-code represents the evaluation strategy of this *rule-combining algorithm*.

```
5264
       Decision permitOverridesRuleCombiningAlgorithm(Rule rule[])
5265
5266
          Boolean atLeastOneError = false;
5267
          Boolean potentialPermit = false;
5268
          Boolean atLeastOneDeny = false;
5269
          for( i=0 ; i < lengthOf(rule) ; i++ )</pre>
5270
5271
             Decision decision = evaluate(rule[i]);
5272
             if (decision == Deny)
5273
5274
                atLeastOneDeny = true;
5275
                continue;
5276
5277
             if (decision == Permit)
```

access_control-xacml-2.0-core-spec-os

```
5278
5279
                return Permit;
5280
5281
             if (decision == NotApplicable)
5282
5283
                continue;
5284
5285
             if (decision == Indeterminate)
5286
5287
                atLeastOneError = true;
5288
5289
                if (effect(rule[i]) == Permit)
5290
5291
                   potentialPermit = true;
5292
5293
                continue;
5294
5295
5296
          if (potentialPermit)
5297
          {
5298
             return Indeterminate;
5299
5300
          if (atLeastOneDeny)
5301
          {
5302
             return Deny;
5303
5304
          if (atLeastOneError)
5305
          {
5306
             return Indeterminate;
5307
5308
          return NotApplicable;
5309
```

The following specification defines the "Permit-overrides" *policy-combining algorithm* of a *policy set*.

In the entire set of *policies* in the *policy set*, if any *policy* evaluates to "Permit", then the result of the *policy* combination SHALL be "Permit". In other words, "Permit" takes precedence, regardless of the result of evaluating any of the other *policies* in the *policy set*. If all *policies* are found to be "NotApplicable" to the *decision request*, then the *policy set* SHALL evaluate to "NotApplicable".

If an error occurs while evaluating the *target* of a *policy*, a reference to a *policy* is considered invalid or the *policy* evaluation results in "Indeterminate", then the *policy set* SHALL evaluate to "Indeterminate", with the appropriate error status, provided no other *policies* evaluate to "Permit" or "Deny".

The following pseudo-code represents the evaluation strategy of this *policy-combining algorithm*.

```
5322
       Decision permitOverridesPolicyCombiningAlgorithm(Policy policy[])
5323
5324
          Boolean atLeastOneError = false;
5325
          Boolean atLeastOneDeny = false;
5326
          for( i=0 ; i < lengthOf(policy) ; i++ )</pre>
5327
5328
             Decision decision = evaluate(policy[i]);
5329
             if (decision == Deny)
5330
5331
                atLeastOneDeny = true;
5332
                continue;
5333
5334
             if (decision == Permit)
```

```
5335
5336
                return Permit;
5337
5338
             if (decision == NotApplicable)
5339
5340
                continue;
5341
5342
             if (decision == Indeterminate)
5343
5344
                atLeastOneError = true;
5345
                continue;
5346
5347
5348
          if (atLeastOneDeny)
5349
5350
             return Deny;
5351
5352
          if (atLeastOneError)
5353
5354
             return Indeterminate;
5355
5356
          return NotApplicable;
5357
```

Obligations of the individual **policies** shall be combined as described in Section 7.14.

C.4. Ordered-permit-overrides

The following specification defines the "Ordered-permit-overrides" *rule-combining algorithm* of a *policy*.

The behavior of this algorithm is identical to that of the Permit-overrides *rule-combining algorithm* with one exception. The order in which the collection of *rules* is evaluated SHALL match the order as listed in the *policy*.

The following specification defines the "Ordered-permit-overrides" *policy-combining algorithm* of a *policy set*.

The behavior of this algorithm is identical to that of the Permit-overrides *policy-combining algorithm* with one exception. The order in which the collection of *policies* is evaluated SHALL match the order as listed in the *policy set*.

C.5. First-applicable

The following specification defines the "First-Applicable " rule-combining algorithm of a policy.

Each *rule* SHALL be evaluated in the order in which it is listed in the *policy*. For a particular *rule*, if the *target* matches and the *condition* evaluates to "True", then the evaluation of the *policy* SHALL halt and the corresponding *effect* of the *rule* SHALL be the result of the evaluation of the *policy* (i.e. "Permit" or "Deny"). For a particular *rule* selected in the evaluation, if the *target* evaluates to "False" or the *condition* evaluates to "False", then the next *rule* in the order SHALL be evaluated. If no further *rule* in the order exists, then the *policy* SHALL evaluate to "NotApplicable".

If an error occurs while evaluating the *target* or *condition* of a *rule*, then the evaluation SHALL halt, and the *policy* shall evaluate to "Indeterminate", with the appropriate error status.

The following pseudo-code represents the evaluation strategy of this *rule-combining algorithm*.

```
5384
5385
       Decision firstApplicableEffectRuleCombiningAlgorithm(Rule rule[])
5386
5387
          for( i = 0 ; i < lengthOf(rule) ; i++ )
5388
5389
             Decision decision = evaluate(rule[i]);
5390
             if (decision == Deny)
5391
             {
5392
                return Deny;
5393
5394
             if (decision == Permit)
5395
             {
5396
                return Permit;
5397
5398
             if (decision == NotApplicable)
5399
             {
5400
                continue;
5401
5402
             if (decision == Indeterminate)
5403
5404
                return Indeterminate;
5405
5406
5407
          return NotApplicable;
5408
```

The following specification defines the "First-applicable" *policy-combining algorithm* of a *policy set*.

Each *policy* is evaluated in the order that it appears in the *policy set*. For a particular *policy*, if the *target* evaluates to "True" and the *policy* evaluates to a determinate value of "Permit" or "Deny", then the evaluation SHALL halt and the *policy set* SHALL evaluate to the *effect* value of that *policy*. For a particular *policy*, if the *target* evaluate to "False", or the *policy* evaluates to "NotApplicable", then the next *policy* in the order SHALL be evaluated. If no further *policy* exists in the order, then the *policy set* SHALL evaluate to "NotApplicable".

If an error were to occur when evaluating the *target*, or when evaluating a specific *policy*, the reference to the *policy* is considered invalid, or the *policy* itself evaluates to "Indeterminate", then the evaluation of the *policy-combining algorithm* shall halt, and the *policy set* shall evaluate to "Indeterminate" with an appropriate error status.

The following pseudo-code represents the evaluation strategy of this *policy-combination algorithm*.

```
Decision firstApplicableEffectPolicyCombiningAlgorithm(Policy policy[])
{
   for( i = 0 ; i < lengthOf(policy) ; i++ )
   {
      Decision decision = evaluate(policy[i]);
      if(decision == Deny)
      {
        return Deny;
   }
}</pre>
```

```
5432
5433
                if(decision == Permit)
5434
                {
5435
                    return Permit;
5436
5437
                if (decision == NotApplicable)
5438
                {
5439
                    continue;
5440
5441
                if (decision == Indeterminate)
5442
5443
                    return Indeterminate;
5444
5445
5446
           return NotApplicable;
5447
```

Obligations of the individual **policies** shall be combined as described in Section 7.14.

C.6. Only-one-applicable

5448

5449

5450

5451

5452

5453

5454

5455

5456

5457

5458

5459

5460

5461

The following specification defines the "Only-one-applicable" *policy-combining algorithm* of a *policy set*.

In the entire set of *policies* in the *policy set*, if no *policy* is considered applicable by virtue of its *target*, then the result of the *policy* combination algorithm SHALL be "NotApplicable". If more than one *policy* is considered applicable by virtue of its *target*, then the result of the *policy* combination algorithm SHALL be "Indeterminate".

If only one **policy** is considered applicable by evaluation of its **target**, then the result of the **policy-combining algorithm** SHALL be the result of evaluating the **policy**.

If an error occurs while evaluating the *target* of a *policy*, or a reference to a *policy* is considered invalid or the *policy* evaluation results in "Indeterminate, then the *policy set* SHALL evaluate to "Indeterminate", with the appropriate error status.

The following pseudo-code represents the evaluation strategy of this policy combining algorithm.

```
5462
       Decision onlyOneApplicablePolicyPolicyCombiningAlogrithm(Policy policy[])
5463
5464
         Boolean
                                          = false;
                           atLeastOne
5465
         Policy
                           selectedPolicy = null;
5466
         ApplicableResult appResult;
5467
5468
         for ( i = 0; i < lengthOf(policy); i++)
5469
5470
            appResult = isApplicable(policy[I]);
5471
5472
            if ( appResult == Indeterminate )
5473
             {
5474
                return Indeterminate;
5475
5476
            if( appResult == Applicable )
5477
5478
                 if ( atLeastOne )
5479
5480
                     return Indeterminate;
5481
5482
                 else
```

```
5483
5484
                    atLeastOne
                                 = true;
5485
                    selectedPolicy = policy[i];
5486
5487
5488
            if ( appResult == NotApplicable )
5489
            {
5490
                continue;
5491
5492
5493
         if ( atLeastOne )
5494
5495
             return evaluate(selectedPolicy);
5496
         }
5497
         else
5498
5499
             return NotApplicable;
5500
         }
5501
```

Appendix D. Acknowledgments

5504 The following individuals contributed to the development of the specification:

5506 Anthony Nadalin 5507 Bill Parducci **Daniel Engovatov** 5508 5509 Don Flinn Ed Coyne 5510 Ernesto Damiani 5511 Frank Siebenlist 5512 **Gerald Brose** 5513 5514 Hal Lockhart 5515 Haruyuki Kawabe James MacLean 5516 5517 John Merrells Ken Yagen 5518

5503

5505

5519 Konstantin Beznosov
5520 Michiharu Kudo
5521 Michael McIntosh
5522 Pierangela Samarati

5523 Pirasenna Velandai Thiyagarajan

Anne Anderson

Polar Humenn 5524 Rebekah Metz 5525 Ron Jacobson 5526 Satoshi Hada 5527 5528 Sekhar Vajjhala Seth Proctor 5529 Simon Godik 5530 5531 Steve Anderson Steve Crocker 5532 Suresh Damodaran 5533

5534 Tim Moses 5535 Von Welch

Appendix E. Notices 5536 5537 OASIS takes no position regarding the validity or scope of any intellectual property or other rights 5538 that might be claimed to pertain to the implementation or use of the technology described in this 5539 document or the extent to which any license under such rights might or might not be available; 5540 neither does it represent that it has made any effort to identify any such rights. Information on 5541 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS 5542 website. Copies of claims of rights made available for publication and any assurances of licenses to 5543 be made available, or the result of an attempt made to obtain a general license or permission for 5544 the use of such proprietary rights by implementors or users of this specification, can be obtained 5545 from the OASIS Executive Director. 5546 OASIS invites any interested party to bring to its attention any copyrights, patents or patent 5547 applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director. 5548 5549 Copyright © OASIS Open 2004-2005. All Rights Reserved. 5550 This document and translations of it may be copied and furnished to others, and derivative works 5551 that comment on or otherwise explain it or assist in its implementation may be prepared, copied. 5552 published and distributed, in whole or in part, without restriction of any kind, provided that the above 5553 copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself does not be modified in any way, such as by removing the copyright notice or 5554 references to OASIS, except as needed for the purpose of developing OASIS specifications, in 5555 5556 which case the procedures for copyrights defined in the OASIS Intellectual Property Rights 5557 document must be followed, or as required to translate it into languages other than English. 5558 The limited permissions granted above are perpetual and will not be revoked by OASIS or its 5559 successors or assigns. This document and the information contained herein is provided on an "AS" 5560 IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL 5561 5562 NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR

FITNESS FOR A PARTICULAR PURPOSE.