



# eXtensible Access Control Markup Language (XACML) Version 3.0 **Plus Errata 01 (redlined)**

**OASIS Standard incorporating Public Review Draft 01 of Errata 01**

**16 February 2017**

## Specification URIs

This version:

<http://docs.oasis-open.org/xacml/3.0/errata01/csprd01/xacml-3.0-core-spec-errata01-csprd01-redlined.doc> (Authoritative)  
<http://docs.oasis-open.org/xacml/3.0/errata01/csprd01/xacml-3.0-core-spec-errata01-csprd01-redlined.html>  
<http://docs.oasis-open.org/xacml/3.0/errata01/csprd01/xacml-3.0-core-spec-errata01-csprd01-redlined.pdf>

Previous version:

N/A

Latest version:

<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.doc> (Authoritative)  
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html>  
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.pdf>

Technical Committee:

OASIS eXtensible Access Control Markup Language (XACML) TC

Chairs:

Bill Parducci ([bill@parducci.net](mailto:bill@parducci.net)), Individual  
Hal Lockhart ([hal.lockhart@oracle.com](mailto:hal.lockhart@oracle.com)), Oracle

Editor:

Erik Rissanen ([erik@axiomatics.com](mailto:erik@axiomatics.com)), Axiomatics AB

Additional artifacts:

This prose specification is one component of a Work Product that also includes:

- List of Errata: *eXtensible Access Control Markup Language (XACML) Version 3.0 Errata 01*. Committee Specification Draft 01 / Public Review Draft 01. <http://docs.oasis-open.org/xacml/3.0/errata01/csprd01/xacml-3.0-core-spec-errata01-csprd01.html>.
- *eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01*. OASIS Standard incorporating Public Review Draft 01 of Errata 01: <http://docs.oasis-open.org/xacml/3.0/errata01/csprd01/xacml-3.0-core-spec-errata01-csprd01-complete.html>.
- XML schema – unmodified from OASIS Standard: <http://docs.oasis-open.org/xacml/3.0/errata01/csprd01/schema/xacml-core-v3-schema-wd-17.xsd>.

Related work:

This specification provides Errata for:

- *eXtensible Access Control Markup Language (XACML) Version 3.0*. Edited by Erik Rissanen. 22 January 2013. OASIS Standard. <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html>.

**Declared XML namespace:**

- urn:oasis:names:tc:xacml:3.0:core:schema:wd-17

**Abstract:**

This document represents the OASIS Standard *eXtensible Access Control Markup Language (XACML) Version 3.0* with markings to indicate the Errata changes.

**Status:**

This document was last revised or approved by the OASIS eXtensible Access Control Markup Language (XACML) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at [https://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=xacml#technical](https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#technical).

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at <https://www.oasis-open.org/committees/xacml/>.

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 TC's web page (<https://www.oasis-open.org/committees/xacml/ipr.php>).

Note that any machine-readable content ([Computer Language Definitions](#)) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

**Citation format:**

When referencing this specification the following citation format should be used:

**[XACML-v3.0-Errata01-redlined]**

*eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01 (redlined)*. Edited by Erik Rissanen. 16 February 2017. OASIS Standard incorporating Public Review Draft 01 of Errata 01. <http://docs.oasis-open.org/xacml/3.0/errata01/csprd01/xacml-3.0-core-spec-errata01-csprd01-redlined.html>. Latest version: <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html>.

---

## Notices

Copyright © OASIS Open 2017. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](#) may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

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

This document and the information contained herein is provided on an "AS 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 NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of [OASIS](#), the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <https://www.oasis-open.org/policies-guidelines/trademark> for above guidance.

---

## Table of Contents

1	Introduction .....	9
1.1	Glossary (non-normative) .....	9
1.1.1	Preferred terms .....	9
1.1.2	Related terms .....	11
1.2	Terminology .....	11
1.3	Schema organization and namespaces .....	12
1.4	Normative References .....	12
1.5	Non-Normative References .....	13
2	Background (non-normative) .....	14
2.1	Requirements .....	14
2.2	Rule and policy combining .....	15
2.3	Combining algorithms .....	15
2.4	Multiple subjects .....	16
2.5	Policies based on subject and resource attributes .....	16
2.6	Multi-valued attributes .....	16
2.7	Policies based on resource contents .....	16
2.8	Operators .....	17
2.9	Policy distribution .....	17
2.10	Policy indexing .....	17
2.11	Abstraction layer .....	18
2.12	Actions performed in conjunction with enforcement .....	18
2.13	Supplemental information about a decision .....	18
3	Models (non-normative) .....	19
3.1	Data-flow model .....	19
3.2	XACML context .....	20
3.3	Policy language model .....	21
3.3.1	Rule .....	21
3.3.2	Policy .....	22
3.3.3	Policy set .....	24
4	Examples (non-normative) .....	25
4.1	Example one .....	25
4.1.1	Example policy .....	25
4.1.2	Example request context .....	26
4.1.3	Example response context .....	28
4.2	Example two .....	28
4.2.1	Example medical record instance .....	28
4.2.2	Example request context .....	29
4.2.3	Example plain-language rules .....	31
4.2.4	Example XACML rule instances .....	31
5	Syntax (normative, with the exception of the schema fragments) .....	43
5.1	Element <PolicySet> .....	43
5.2	Element <Description> .....	45
5.3	Element <PolicyIssuer> .....	45

5.4 Element <PolicySetDefaults>	45
5.5 Element <XPathVersion>	46
5.6 Element <Target>	46
5.7 Element <AnyOf>	46
5.8 Element <AllOf>	47
5.9 Element <Match>	47
5.10 Element <PolicySetIdReference>	48
5.11 Element <PolicyIdReference>	48
5.12 Simple type VersionType	48
5.13 Simple type VersionMatchType	49
5.14 Element <Policy>	49
5.15 Element <PolicyDefaults>	51
5.16 Element <CombinerParameters>	51
5.17 Element <CombinerParameter>	52
5.18 Element <RuleCombinerParameters>	52
5.19 Element <PolicyCombinerParameters>	53
5.20 Element <PolicySetCombinerParameters>	53
5.21 Element <Rule>	54
5.22 Simple type EffectType	55
5.23 Element <VariableDefinition>	55
5.24 Element <VariableReference>	55
5.25 Element <Expression>	56
5.26 Element <Condition>	56
5.27 Element <Apply>	56
5.28 Element <Function>	57
5.29 Element <AttributeDesignator>	57
5.30 Element <AttributeSelector>	59
5.31 Element <AttributeValue>	60
5.32 Element <Obligations>	60
5.33 Element <AssociatedAdvice>	60
5.34 Element <Obligation>	61
5.35 Element <Advice>	61
5.36 Element <AttributeAssignment>	61
5.37 Element <ObligationExpressions>	62
5.38 Element <AdviceExpressions>	62
5.39 Element <ObligationExpression>	63
5.40 Element <AdviceExpression>	63
5.41 Element <AttributeAssignmentExpression>	64
5.42 Element <Request>	65
5.43 Element <RequestDefaults>	66
5.44 Element <Attributes>	66
5.45 Element <Content>	67
5.46 Element <Attribute>	67
5.47 Element <Response>	67
5.48 Element <Result>	68

5.49	Element <PolicyIdentifierList>	69
5.50	Element <MultiRequests>	69
5.51	Element <RequestReference>	70
5.52	Element <AttributesReference>	70
5.53	Element <Decision>	70
5.54	Element <Status>	71
5.55	Element <StatusCode>	71
5.56	Element <StatusMessage>	71
5.57	Element <StatusDetail>	72
5.58	Element <MissingAttributeDetail>	72
6	XPath 2.0 definitions	74
7	Functional requirements	76
7.1	Unicode issues	76
7.1.1	Normalization	76
7.1.2	Version of Unicode	76
7.2	Policy enforcement point	76
7.2.1	Base PEP	76
7.2.2	Deny-biased PEP	76
7.2.3	Permit-biased PEP	77
7.3	Attribute evaluation	77
7.3.1	Structured attributes	77
7.3.2	Attribute bags	77
7.3.3	Multivalued attributes	78
7.3.4	Attribute Matching	78
7.3.5	Attribute Retrieval	78
7.3.6	Environment Attributes	79
7.3.7	AttributeSelector evaluation	79
7.4	Expression evaluation	80
7.5	Arithmetic evaluation	80
7.6	Match evaluation	80
7.7	Target evaluation	82
7.8	VariableReference Evaluation	82
7.9	Condition evaluation	83
7.10	Extended Indeterminate	83
7.11	Rule evaluation	83
7.12	Policy evaluation	83
7.13	Policy Set evaluation	84
7.14	Policy and Policy set value for Indeterminate Target	84
7.15	PolicySetIdReference and PolicyIdReference evaluation	85
7.16	Hierarchical resources	85
7.17	Authorization decision	85
7.18	Obligations and advice	85
7.19	Exception handling	86
7.19.1	Unsupported functionality	86
7.19.2	Syntax and type errors	86

7.19.3 Missing attributes.....	86
7.20 Identifier equality .....	86
8 XACML extensibility points (non-normative).....	88
8.1 Extensible XML attribute types.....	88
8.2 Structured attributes .....	88
9 Security and privacy considerations (non-normative) .....	89
9.1 Threat model .....	89
9.1.1 Unauthorized disclosure .....	89
9.1.2 Message replay .....	89
9.1.3 Message insertion .....	89
9.1.4 Message deletion .....	90
9.1.5 Message modification .....	90
9.1.6 NotApplicable results .....	90
9.1.7 Negative rules .....	90
9.1.8 Denial of service.....	91
9.2 Safeguards .....	91
9.2.1 Authentication .....	91
9.2.2 Policy administration.....	91
9.2.3 Confidentiality.....	92
9.2.4 Policy integrity.....	92
9.2.5 Policy identifiers .....	92
9.2.6 Trust model .....	93
9.2.7 Privacy.....	93
9.3 Unicode security issues .....	94
9.4 Identifier equality .....	94
10 Conformance .....	95
10.1 Introduction .....	95
10.2 Conformance tables.....	95
10.2.1 Schema elements.....	95
10.2.2 Identifier Prefixes.....	96
10.2.3 Algorithms .....	96
10.2.4 Status Codes.....	97
10.2.5 Attributes .....	97
10.2.6 Identifiers.....	97
10.2.7 Data-types.....	98
10.2.8 Functions.....	98
10.2.9 Identifiers planned for future deprecation .....	103
Appendix A. Data-types and functions (normative).....	105
A.1 Introduction .....	105
A.2 Data-types .....	105
A.3 Functions .....	107
A.3.1 Equality predicates.....	107
A.3.2 Arithmetic functions.....	109
A.3.3 String conversion functions .....	110
A.3.4 Numeric data-type conversion functions .....	110

A.3.5 Logical functions .....	110
A.3.6 Numeric comparison functions .....	111
A.3.7 Date and time arithmetic functions.....	111
A.3.8 Non-numeric comparison functions .....	112
A.3.9 String functions .....	115
A.3.10 Bag functions .....	119
A.3.11 Set functions .....	120
A.3.12 Higher-order bag functions .....	120
A.3.13 Regular-expression-based functions.....	124
A.3.14 Special match functions.....	126
A.3.15 XPath-based functions .....	126
A.3.16 Other functions.....	127
A.3.17 Extension functions and primitive types .....	127
A.4 Functions, data types, attributes and algorithms planned for deprecation .....	128
Appendix B. XACML identifiers (normative) .....	130
B.1 XACML namespaces.....	130
B.2 Attribute categories .....	130
B.3 Data-types .....	130
B.4 Subject attributes .....	131
B.5 Resource attributes .....	132
B.6 Action attributes .....	132
B.7 Environment attributes .....	132
B.8 Status codes .....	133
B.9 Combining algorithms.....	133
Appendix C. Combining algorithms (normative) .....	135
C.1 Extended Indeterminate values .....	135
C.2 Deny-overrides .....	135
C.3 Ordered-deny-overrides .....	137
C.4 Permit-overrides.....	137
C.5 Ordered-permit-overrides .....	138
C.6 Deny-unless-permit.....	139
C.7 Permit-unless-deny .....	139
C.8 First-applicable .....	140
C.9 Only-one-applicable .....	142
C.10 Legacy Deny-overrides .....	143
C.11 Legacy Ordered-deny-overrides .....	144
C.12 Legacy Permit-overrides .....	145
C.13 Legacy Ordered-permit-overrides .....	147
Appendix D. Acknowledgements .....	148
Appendix E. Revision History .....	149



---

1	<b>1 Introduction</b>
2	<b>1.1 Glossary (non-normative)</b>
3	<b>1.1.1 Preferred terms</b>
4	<b>Access</b>
5	Performing an <i>action</i>
6	<b>Access control</b>
7	Controlling <i>access</i> in accordance with a <i>policy</i> or <i>policy set</i>
8	<b>Action</b>
9	An operation on a <i>resource</i>
10	<b>Advice</b>
11	A supplementary piece of information in a <i>policy</i> or <i>policy set</i> which is provided to the <i>PEP</i> with
12	the <i>decision</i> of the <i>PDP</i> .
13	<b>Applicable policy</b>
14	The set of <i>policies</i> and <i>policy sets</i> that governs <i>access</i> for a specific <i>decision request</i>
15	<b>Attribute</b>
16	Characteristic of a <i>subject</i> , <i>resource</i> , <i>action</i> or <i>environment</i> that may be referenced in a
17	<i>predicate</i> or <i>target</i> (see also – <i>named attribute</i> )
18	<b>Authorization decision</b>
19	The result of evaluating <i>applicable policy</i> , returned by the <i>PDP</i> to the <i>PEP</i> . A function that
20	evaluates to "Permit", "Deny", "Indeterminate" or "NotApplicable", and (optionally) a set of
21	<i>obligations and advice</i>
22	<b>Bag</b>
23	An unordered collection of values, in which there may be duplicate values
24	<b>Condition</b>
25	An expression of <i>predicates</i> . A function that evaluates to "True", "False" or "Indeterminate"
26	<b>Conjunctive sequence</b>
27	A sequence of <i>predicates</i> combined using the logical 'AND' operation
28	<b>Context</b>
29	The canonical representation of a <i>decision request</i> and an <i>authorization decision</i>
30	<b>Context handler</b>
31	The system entity that converts <i>decision requests</i> in the native request format to the XACML
32	canonical form, coordinates with Policy Information Points to add attribute values to the request
33	context, and converts <i>authorization decisions</i> in the XACML canonical form to the native
34	response format
35	<b>Decision</b>
36	The result of evaluating a <i>rule</i> , <i>policy</i> or <i>policy set</i>
37	<b>Decision request</b>
38	The request by a <i>PEP</i> to a <i>PDP</i> to render an <i>authorization decision</i>

- 39 **Disjunctive sequence**  
40 A sequence of *predicates* combined using the logical 'OR' operation
- 41 **Effect**  
42 The intended consequence of a satisfied *rule* (either "Permit" or "Deny")
- 43 **Environment**  
44 The set of *attributes* that are relevant to an *authorization decision* and are independent of a  
45 particular *subject*, *resource* or *action*
- 46 **Identifier equality**  
47 The identifier equality operation which is defined in section 7.20.
- 48 **Issuer**  
49 A set of *attributes* describing the source of a *policy*
- 50 **Named attribute**  
51 A specific instance of an *attribute*, determined by the *attribute* name and type, the identity of the  
52 *attribute* holder (which may be of type: *subject*, *resource*, *action* or *environment*) and  
53 (optionally) the identity of the issuing authority
- 54 **Obligation**  
55 An operation specified in a *rule*, *policy* or *policy set* that should be performed by the *PEP* in  
56 conjunction with the enforcement of an *authorization decision*
- 57 **Policy**  
58 A set of *rules*, an identifier for the *rule-combining algorithm* and (optionally) a set of  
59 *obligations* or *advice*. May be a component of a *policy set*
- 60 **Policy administration point (PAP)**  
61 The system entity that creates a *policy* or *policy set*
- 62 **Policy-combining algorithm**  
63 The procedure for combining the *decision* and *obligations* from multiple *policies*
- 64 **Policy decision point (PDP)**  
65 The system entity that evaluates *applicable policy* and renders an *authorization decision*.  
66 This term is defined in a joint effort by the IETF Policy Framework Working Group and the  
67 Distributed Management Task Force (DMTF)/Common Information Model (CIM) in [RFC3198].  
68 This term corresponds to "Access Decision Function" (ADF) in [ISO10181-3].
- 69 **Policy enforcement point (PEP)**  
70 The system entity that performs *access control*, by making *decision requests* and enforcing  
71 *authorization decisions*. This term is defined in a joint effort by the IETF Policy Framework  
72 Working Group and the Distributed Management Task Force (DMTF)/Common Information Model  
73 (CIM) in [RFC3198]. This term corresponds to "Access Enforcement Function" (AEF) in  
74 [ISO10181-3].
- 75 **Policy information point (PIP)**  
76 The system entity that acts as a source of *attribute* values
- 77 **Policy set**  
78 A set of *policies*, other *policy sets*, a *policy-combining algorithm* and (optionally) a set of  
79 *obligations* or *advice*. May be a component of another *policy set*
- 80 **Predicate**  
81 A statement about *attributes* whose truth can be evaluated
- 82 **Resource**

83 Data, service or system component

84 **Rule**

85 A *target*, an *effect*, a *condition* and (optionally) a set of *obligations* or *advice*. A component of  
86 a *policy*

87 **Rule-combining algorithm**

88 The procedure for combining *decisions* from multiple *rules*

89 **Subject**

90 An actor whose *attributes* may be referenced by a *predicate*

91 **Target**

92 ~~An element of an XACML rule, policy, or policy set which matches specified values of resource,~~  
93 ~~subject, environment, action, or other custom attributes against those provided in the request~~  
94 ~~context as a part of the process of determining whether the rule, policy, or policy set is applicable~~  
95 ~~to the current decision.~~

96 **Type Unification**

97 The method by which two type expressions are "unified". The type expressions are matched  
98 along their structure. Where a type variable appears in one expression it is then "unified" to  
99 represent the corresponding structure element of the other expression, be it another variable or  
100 subexpression. All variable assignments must remain consistent in both structures. Unification  
101 fails if the two expressions cannot be aligned, either by having dissimilar structure, or by having  
102 instance conflicts, such as a variable needs to represent both "xs:string" and "xs:integer". For a  
103 full explanation of *type unification*, please see [Hancock].

104 **1.1.2 Related terms**

105 In the field of *access control* and authorization there are several closely related terms in common use.  
106 For purposes of precision and clarity, certain of these terms are not used in this specification.  
107 For instance, the term *attribute* is used in place of the terms: group and role.  
108 In place of the terms: privilege, permission, authorization, entitlement and right, we use the term *rule*.  
109 The term object is also in common use, but we use the term *resource* in this specification.  
110 Requestors and initiators are covered by the term *subject*.

111 **1.2 Terminology**

112 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD  
113 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described  
114 in [RFC2119].

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

118 Listings of XACML schema appear like this.

120 Example code listings appear like this.

122 Conventional XML namespace prefixes are used throughout the listings in this specification to stand for  
123 their respective namespaces as follows, whether or not a namespace declaration is present in the  
124 example:

- 125 • The prefix `xacml`: stands for the XACML 3.0 namespace.
- 126 • The prefix `ds`: stands for the W3C XML Signature namespace [DS].

Deleted: The set  
Deleted: decision requests, identified by definitions for  
Formatted: Default Paragraph Font  
Formatted: Default Paragraph Font  
Deleted: and  
Formatted: Default Paragraph Font  
Deleted: that a  
Formatted: Default Paragraph Font  
Formatted: Default Paragraph Font  
Formatted: Default Paragraph Font  
Deleted: intended  
Deleted: evaluate

- 133 • The prefix `xs`: stands for the W3C XML Schema namespace **[XS]**.
  - 134 • The prefix `xf`: stands for the XQuery 1.0 and XPath 2.0 Function and Operators specification
  - 135 namespace **[XF]**.
  - 136 • The prefix `xml`: stands for the XML namespace <http://www.w3.org/XML/1998/namespace>.
- 137 This specification uses the following typographical conventions in text: `<XACMLElement>`,  
 138 `<ns:ForeignElement>`, `Attribute`, `Datatype`, `OtherCode`. Terms in ***bold-face italic*** are intended  
 139 to have the meaning defined in the Glossary.

### 140 1.3 Schema organization and namespaces

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

142 `urn:oasis:names:tc:xacml:3.0:core:schema:wd-17`

### 143 1.4 Normative References

- 144 **[CMF]** Martin J. Dürst et al, eds., *Character Model for the World Wide Web 1.0:*
- 145 *Fundamentals*, W3C Recommendation 15 February 2005,
- 146 <http://www.w3.org/TR/2005/REC-charmod-20050215/>
- 147 **[DS]** D. Eastlake et al., *XML-Signature Syntax and Processing*,
- 148 <http://www.w3.org/TR/xmldsig-core/>, World Wide Web Consortium.
- 149 **[exc-c14n]** J. Boyer et al, eds., *Exclusive XML Canonicalization, Version 1.0*, W3C
- 150 Recommendation 18 July 2002, [http://www.w3.org/TR/2002/REC-xml-exc-c14n-](http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/)
- 151 [20020718/](http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/)
- 152 **[Hancock]** Hancock, *Polymorphic Type Checking*, in Simon L. Peyton Jones,
- 153 *Implementation of Functional Programming Languages*, Section 8,
- 154 Prentice-Hall International, 1987.
- 155 **[Hier]** *XACML v3.0 Hierarchical Resource Profile Version 1.0*. 11 March 2010.
- 156 Committee Specification Draft 03. [http://docs.oasis-open.org/xacml/3.0/xacml-](http://docs.oasis-open.org/xacml/3.0/xacml-3.0-hierarchical-v1-spec-cd-03-en.html)
- 157 [3.0-hierarchical-v1-spec-cd-03-en.html](http://docs.oasis-open.org/xacml/3.0/xacml-3.0-hierarchical-v1-spec-cd-03-en.html)
- 158 **[IEEE754]** IEEE Standard for Binary Floating-Point Arithmetic 1985, ISBN 1-5593-7653-8,
- 159 IEEE Product No. SH10116-TBR.
- 160 **[INFOSET]** [XML Information Set \(Second Edition\), W3C Recommendation](https://www.w3.org/TR/xml-infoset/)
- 161 [4 February 2004, available at https://www.w3.org/TR/xml-infoset/](https://www.w3.org/TR/xml-infoset/)
- 162 **[ISO10181-3]** ISO/IEC 10181-3:1996 Information technology – Open Systems Interconnection -
- 163 - Security frameworks for open systems: Access control framework.
- 164 **[Kudo00]** Kudo M and Hada S, *XML document security based on provisional authorization*,
- 165 Proceedings of the Seventh ACM Conference on Computer and Communications
- 166 Security, Nov 2000, Athens, Greece, pp 87-96.
- 167 **[LDAP-1]** RFC2256, *A summary of the X500(96) User Schema for use with LDAPv3*,
- 168 Section 5, M Wahl, December 1997, <http://www.ietf.org/rfc/rfc2256.txt>
- 169 **[LDAP-2]** RFC2798, *Definition of the inetOrgPerson*, M. Smith, April 2000
- 170 <http://www.ietf.org/rfc/rfc2798.txt>
- 171 **[MathML]** *Mathematical Markup Language (MathML)*, Version 2.0, W3C Recommendation,
- 172 21 October 2003. Available at: [http://www.w3.org/TR/2003/REC-MathML2-](http://www.w3.org/TR/2003/REC-MathML2-20031021/)
- 173 [20031021/](http://www.w3.org/TR/2003/REC-MathML2-20031021/)
- 174 **[Multi]** OASIS Committee Draft 03, *XACML v3.0 Multiple Decision Profile Version 1.0*,
- 175 11 March 2010, [http://docs.oasis-open.org/xacml/3.0/xacml-3.0-multiple-v1-spec-](http://docs.oasis-open.org/xacml/3.0/xacml-3.0-multiple-v1-spec-cd-03-en.doc)
- 176 [cd-03-en.doc](http://docs.oasis-open.org/xacml/3.0/xacml-3.0-multiple-v1-spec-cd-03-en.doc)
- 177 **[Perritt93]** Perritt, H. Knowbots, *Permissions Headers and Contract Law*, Conference on
- 178 Technological Strategies for Protecting Intellectual Property in the Networked
- 179 Multimedia Environment, April 1993. Available at:
- 180 <http://www.ifla.org/documents/infopol/copyright/perh2.txt>

181	<b>[RBAC]</b>	David Ferraiolo and Richard Kuhn, <i>Role-Based Access Controls</i> , 15th National Computer Security Conference, 1992.
182		
183	<b>[RFC2119]</b>	S. Bradner, <i>Key words for use in RFCs to Indicate Requirement Levels</i> , <a href="http://www.ietf.org/rfc/rfc2119.txt">http://www.ietf.org/rfc/rfc2119.txt</a> , IETF RFC 2119, March 1997.
184		
185	<b>[RFC2396]</b>	Berners-Lee T, Fielding R, Masinter L, <i>Uniform Resource Identifiers (URI): Generic Syntax</i> . Available at: <a href="http://www.ietf.org/rfc/rfc2396.txt">http://www.ietf.org/rfc/rfc2396.txt</a>
186		
187	<b>[RFC2732]</b>	Hinden R, Carpenter B, Masinter L, <i>Format for Literal IPv6 Addresses in URL's</i> . Available at: <a href="http://www.ietf.org/rfc/rfc2732.txt">http://www.ietf.org/rfc/rfc2732.txt</a>
188		
189	<b>[RFC3198]</b>	IETF RFC 3198: <i>Terminology for Policy-Based Management</i> , November 2001. <a href="http://www.ietf.org/rfc/rfc3198.txt">http://www.ietf.org/rfc/rfc3198.txt</a>
190		
191	<b>[UAX15]</b>	Mark Davis, Martin Dürst, <i>Unicode Standard Annex #15: Unicode Normalization Forms, Unicode 5.1</i> , available from <a href="http://unicode.org/reports/tr15/">http://unicode.org/reports/tr15/</a>
192		
193	<b>[UTR36]</b>	Davis, Mark, Suignard, Michel, <i>Unicode Technocal Report #36: Unicode Security Considerations</i> . Available at <a href="http://www.unicode.org/reports/tr36/">http://www.unicode.org/reports/tr36/</a>
194		
195	<b>[XACMLAdmin]</b>	OASIS Committee Draft 03, <i>XACML v3.0 Administration and Delegation Profile Version 1.0</i> . 11 March 2010. <a href="http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-cd-03-en.doc">http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-cd-03-en.doc</a>
196		
197		
198	<b>[XACMLv1.0]</b>	OASIS Standard, <i>Extensible access control markup language (XACML) Version 1.0</i> . 18 February 2003. <a href="http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf">http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf</a>
199		
200		
201	<b>[XACMLv1.1]</b>	OASIS Committee Specification, <i>Extensible access control markup language (XACML) Version 1.1</i> . 7 August 2003. <a href="http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf">http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf</a>
202		
203		
204	<b>[XF]</b>	<i>XQuery 1.0 and XPath 2.0 Functions and Operators</i> , W3C Recommendation 23 January 2007. Available at: <a href="http://www.w3.org/TR/2007/REC-xpath-functions-20070123/">http://www.w3.org/TR/2007/REC-xpath-functions-20070123/</a>
205		
206		
207	<b>[XML]</b>	Bray, Tim, et.al. eds, <i>Extensible Markup Language (XML) 1.0 (Fifth Edition)</i> , W3C Recommendation 26 November 2008, available at <a href="http://www.w3.org/TR/2008/REC-xml-20081126/">http://www.w3.org/TR/2008/REC-xml-20081126/</a>
208		
209		
210	<b>[XMLid]</b>	Marsh, Jonathan, et.al. eds, <i>xml:id Version 1.0</i> . W3C Recommendation 9 September 2005. Available at: <a href="http://www.w3.org/TR/2005/REC-xml-id-20050909/">http://www.w3.org/TR/2005/REC-xml-id-20050909/</a>
211		
212		
213	<b>[XS]</b>	<i>XML Schema, parts 1 and 2</i> . Available at: <a href="http://www.w3.org/TR/xmlschema-1/">http://www.w3.org/TR/xmlschema-1/</a> and <a href="http://www.w3.org/TR/xmlschema-2/">http://www.w3.org/TR/xmlschema-2/</a>
214		
215	<b>[XPath]</b>	<i>XML Path Language (XPath), Version 1.0</i> , W3C Recommendation 16 November 1999. Available at: <a href="http://www.w3.org/TR/xpath">http://www.w3.org/TR/xpath</a>
216		
217	<b>[XPathFunc]</b>	<i>XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition)</i> , W3C Recommendation 14 December 2010. Available at: <a href="http://www.w3.org/TR/2010/REC-xpath-functions-20101214/">http://www.w3.org/TR/2010/REC-xpath-functions-20101214/</a>
218		
219		
220	<b>[XSLT]</b>	<i>XSL Transformations (XSLT) Version 1.0</i> , W3C Recommendation 16 November 1999. Available at: <a href="http://www.w3.org/TR/xslt">http://www.w3.org/TR/xslt</a>
221		

## 222 1.5 Non-Normative References

223	<b>[CM]</b>	<i>Character model for the World Wide Web 1.0: Normalization</i> , W3C Working Draft, 27 October 2005, <a href="http://www.w3.org/TR/2005/WD-charmod-norm-20051027/">http://www.w3.org/TR/2005/WD-charmod-norm-20051027/</a> , World Wide Web Consortium.
224		
225		
226	<b>[Hinton94]</b>	Hinton, H, M, Lee, E, S, <i>The Compatibility of Policies</i> , Proceedings 2nd ACM Conference on Computer and Communications Security, Nov 1994, Fairfax, Virginia, USA.
227		
228		
229	<b>[Sloman94]</b>	Sloman, M. <i>Policy Driven Management for Distributed Systems</i> . Journal of Network and Systems Management, Volume 2, part 4. Plenum Press. 1994.
230		

## 231 2 Background (non-normative)

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

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

249 For these reasons, there is a pressing need for a common language for expressing security policy. If  
250 implemented throughout an enterprise, a common policy language allows the enterprise to manage the  
251 enforcement of all the elements of its security policy in all the components of its information systems.  
252 Managing security policy may include some or all of the following steps: writing, reviewing, testing,  
253 approving, issuing, combining, analyzing, modifying, withdrawing, retrieving, and enforcing policy.

254 XML is a natural choice as the basis for the common security-policy language, due to the ease with which  
255 its syntax and semantics can be extended to accommodate the unique requirements of this application,  
256 and the widespread support that it enjoys from all the main platform and tool vendors.

### 257 2.1 Requirements

258 The basic requirements of a policy language for expressing information system security policy are:

- 259 • To provide a method for combining individual **rules** and **policies** into a single **policy set** that applies  
260 to a particular **decision request**.
- 261 • To provide a method for flexible definition of the procedure by which **rules** and **policies** are  
262 combined.
- 263 • To provide a method for dealing with multiple **subjects** acting in different capacities.
- 264 • To provide a method for basing an **authorization decision** on **attributes** of the **subject** and  
265 **resource**.
- 266 • To provide a method for dealing with multi-valued **attributes**.
- 267 • To provide a method for basing an **authorization decision** on the contents of an information  
268 **resource**.
- 269 • To provide a set of logical and mathematical operators on **attributes** of the **subject**, **resource** and  
270 **environment**.
- 271 • To provide a method for handling a distributed set of **policy** components, while abstracting the  
272 method for locating, retrieving and authenticating the **policy** components.
- 273 • To provide a method for rapidly identifying the **policy** that applies to a given **action**, based upon the  
274 values of **attributes** of the **subjects**, **resource** and **action**.
- 275 • To provide an abstraction-layer that insulates the **policy**-writer from the details of the application  
276 environment.

277 • To provide a method for specifying a set of **actions** that must be performed in conjunction with **policy**  
278 enforcement.

279 The motivation behind XACML is to express these well-established ideas in the field of **access control**  
280 policy using an extension language of XML. The XACML solutions for each of these requirements are  
281 discussed in the following sections.

## 282 2.2 Rule and policy combining

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

288 XACML defines three top-level **policy** elements: <Rule>, <Policy> and <PolicySet>. The <Rule>  
289 element contains a Boolean expression that can be evaluated in isolation, but that is not intended to be  
290 accessed in isolation by a **PDP**. So, it is not intended to form the basis of an **authorization decision** by  
291 itself. It is intended to exist in isolation only within an XACML **PAP**, where it may form the basic unit of  
292 management.

293 The <Policy> element contains a set of <Rule> elements and a specified procedure for combining the  
294 results of their evaluation. It is the basic unit of **policy** used by the **PDP**, and so it is intended to form the  
295 basis of an **authorization decision**.

296 The <PolicySet> element contains a set of <Policy> or other <PolicySet> elements and a  
297 specified procedure for combining the results of their evaluation. It is the standard means for combining  
298 separate **policies** into a single combined **policy**.

299 Hinton et al [Hinton94] discuss the question of the compatibility of separate **policies** applicable to the  
300 same **decision request**.

## 301 2.3 Combining algorithms

302 XACML defines a number of combining algorithms that can be identified by a `RuleCombiningAlgId` or  
303 `PolicyCombiningAlgId` attribute of the <Policy> or <PolicySet> elements, respectively. The  
304 **rule-combining algorithm** defines a procedure for arriving at an **authorization decision** given the  
305 individual results of evaluation of a set of **rules**. Similarly, the **policy-combining algorithm** defines a  
306 procedure for arriving at an **authorization decision** given the individual results of evaluation of a set of  
307 **policies**. Some examples of standard combining algorithms are (see Appendix C for a full list of standard  
308 combining algorithms):

- 309 • Deny-overrides (Ordered and Unordered),
- 310 • Permit-overrides (Ordered and Unordered),
- 311 • First-applicable and
- 312 • Only-one-applicable.

313 In the case of the Deny-overrides algorithm, if a single <Rule> or <Policy> element is encountered that  
314 evaluates to "Deny", then, regardless of the evaluation result of the other <Rule> or <Policy> elements  
315 in the **applicable policy**, the combined result is "Deny".

316 Likewise, in the case of the Permit-overrides algorithm, if a single "Permit" result is encountered, then the  
317 combined result is "Permit".

318 In the case of the "First-applicable" combining algorithm, the combined result is the same as the result of  
319 evaluating the first <Rule>, <Policy> or <PolicySet> element in the list of **rules** whose **target** and  
320 **condition** is applicable to the **decision request**.

321 The "Only-one-applicable" **policy-combining algorithm** only applies to **policies**. The result of this  
322 combining algorithm ensures that one and only one **policy** or **policy set** is applicable by virtue of their  
323 **targets**. If no **policy** or **policy set** applies, then the result is "NotApplicable", but if more than one **policy**  
324 or **policy set** is applicable, then the result is "Indeterminate". When exactly one **policy** or **policy set** is

325 applicable, the result of the combining algorithm is the result of evaluating the single **applicable policy** or  
326 **policy set**.

327 **Policies** and **policy sets** may take parameters that modify the behavior of the combining algorithms.  
328 However, none of the standard combining algorithms is affected by parameters.

329 Users of this specification may, if necessary, define their own combining algorithms.

## 330 2.4 Multiple subjects

331 **Access control policies** often place requirements on the **actions** of more than one **subject**. For  
332 instance, the **policy** governing the execution of a high-value financial transaction may require the  
333 approval of more than one individual, acting in different capacities. Therefore, XACML recognizes that  
334 there may be more than one **subject** relevant to a **decision request**. Different **attribute** categories are  
335 used to differentiate between **subjects** acting in different capacities. Some standard values for these  
336 **attribute** categories are specified, and users may define additional ones.

## 337 2.5 Policies based on subject and resource attributes

338 Another common requirement is to base an **authorization decision** on some characteristic of the  
339 **subject** other than its identity. Perhaps, the most common application of this idea is the **subject's** role  
340 **[RBAC]**. XACML provides facilities to support this approach. **Attributes of subjects** contained in the  
341 request **context** may be identified by the `<AttributeDesignator>` element. This element contains a  
342 URN that identifies the **attribute**. Alternatively, the `<AttributeSelector>` element may contain an  
343 XPath expression over the `<Content>` element of the **subject** to identify a particular **subject attribute**  
344 value by its location in the **context** (see Section 2.11 for an explanation of **context**).

345 XACML provides a standard way to reference the **attributes** defined in the LDAP series of specifications  
346 **[LDAP-1]**, **[LDAP-2]**. This is intended to encourage implementers to use standard **attribute** identifiers for  
347 some common **subject attributes**.

348 Another common requirement is to base an **authorization decision** on some characteristic of the  
349 **resource** other than its identity. XACML provides facilities to support this approach. **Attributes of the**  
350 **resource** may be identified by the `<AttributeDesignator>` element. This element contains a URN  
351 that identifies the **attribute**. Alternatively, the `<AttributeSelector>` element may contain an XPath  
352 expression over the `<Content>` element of the **resource** to identify a particular **resource attribute** value  
353 by its location in the **context**.

## 354 2.6 Multi-valued attributes

355 The most common techniques for communicating **attributes** (LDAP, XPath, SAML, etc.) support multiple  
356 values per **attribute**. Therefore, when an XACML **PDP** retrieves the value of a **named attribute**, the  
357 result may contain multiple values. A collection of such values is called a **bag**. A **bag** differs from a set in  
358 that it may contain duplicate values, whereas a set may not. Sometimes this situation represents an  
359 error. Sometimes the XACML **rule** is satisfied if any one of the **attribute** values meets the criteria  
360 expressed in the **rule**.

361 XACML provides a set of functions that allow a **policy** writer to be absolutely clear about how the **PDP**  
362 should handle the case of multiple **attribute** values. These are the "higher-order" functions (see Section  
363 A.3).

## 364 2.7 Policies based on resource contents

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

369 XACML provides facilities for doing this when the information **resource** can be represented as an XML  
370 document. The `<AttributeSelector>` element may contain an XPath expression over the



371 <Content> element of the **resource** to identify data in the information **resource** to be used in the **policy**  
372 evaluation.

373 In cases where the information **resource** is not an XML document, specified **attributes** of the **resource**  
374 can be referenced, as described in Section 2.5.

## 375 2.8 Operators

376 Information security **policies** operate upon **attributes** of **subjects**, the **resource**, the **action** and the  
377 **environment** in order to arrive at an **authorization decision**. In the process of arriving at the  
378 **authorization decision**, **attributes** of many different types may have to be compared or computed. For  
379 instance, in a financial application, a person's available credit may have to be calculated by adding their  
380 credit limit to their account balance. The result may then have to be compared with the transaction value.  
381 This sort of situation gives rise to the need for arithmetic operations on **attributes** of the **subject** (account  
382 balance and credit limit) and the **resource** (transaction value).

383 Even more commonly, a **policy** may identify the set of roles that are permitted to perform a particular  
384 **action**. The corresponding operation involves checking whether there is a non-empty intersection  
385 between the set of roles occupied by the **subject** and the set of roles identified in the **policy**; hence the  
386 need for set operations.

387 XACML includes a number of built-in functions and a method of adding non-standard functions. These  
388 functions may be nested to build arbitrarily complex expressions. This is achieved with the <Apply>  
389 element. The <Apply> element has an XML attribute called `FunctionId` that identifies the function to  
390 be applied to the contents of the element. Each standard function is defined for specific argument data-  
391 type combinations, and its return data-type is also specified. Therefore, data-type consistency of the  
392 **policy** can be checked at the time the **policy** is written or parsed. And, the types of the data values  
393 presented in the request **context** can be checked against the values expected by the **policy** to ensure a  
394 predictable outcome.

395 In addition to operators on numerical and set arguments, operators are defined for date, time and  
396 duration arguments.

397 Relationship operators (equality and comparison) are also defined for a number of data-types, including  
398 the RFC822 and X.500 name-forms, strings, URIs, etc.

399 Also noteworthy are the operators over Boolean data-types, which permit the logical combination of  
400 **predicates** in a **rule**. For example, a **rule** may contain the statement that **access** may be permitted  
401 during business hours AND from a terminal on business premises.

402 The XACML method of representing functions borrows from MathML [**MathML**] and from the XQuery 1.0  
403 and XPath 2.0 Functions and Operators specification [**XF**].

## 404 2.9 Policy distribution

405 In a distributed system, individual **policy** statements may be written by several **policy** writers and  
406 enforced at several enforcement points. In addition to facilitating the collection and combination of  
407 independent **policy** components, this approach allows **policies** to be updated as required. XACML  
408 **policy** statements may be distributed in any one of a number of ways. But, XACML does not describe  
409 any normative way to do this. Regardless of the means of distribution, **PDPs** are expected to confirm, by  
410 examining the **policy's** <Target> element that the **policy** is applicable to the **decision request** that it is  
411 processing.

412 <Policy> elements may be attached to the information **resources** to which they apply, as described by  
413 Perritt [**Perritt93**]. Alternatively, <Policy> elements may be maintained in one or more locations from  
414 which they are retrieved for evaluation. In such cases, the **applicable policy** may be referenced by an  
415 identifier or locator closely associated with the information **resource**.

## 416 2.10 Policy indexing

417 For efficiency of evaluation and ease of management, the overall security **policy** in force across an  
418 enterprise may be expressed as multiple independent **policy** components. In this case, it is necessary to

419 identify and retrieve the **applicable policy** statement and verify that it is the correct one for the requested  
420 **action** before evaluating it. This is the purpose of the <Target> element in XACML.

421 Two approaches are supported:

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

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

## 430 2.11 Abstraction layer

431 **PEPs** come in many forms. For instance, a **PEP** may be part of a remote-access gateway, part of a Web  
432 server or part of an email user-agent, etc. It is unrealistic to expect that all **PEPs** in an enterprise do  
433 currently, or will in the future, issue **decision requests** to a **PDP** in a common format. Nevertheless, a  
434 particular **policy** may have to be enforced by multiple **PEPs**. It would be inefficient to force a **policy**  
435 writer to write the same **policy** several different ways in order to accommodate the format requirements of  
436 each **PEP**. Similarly **attributes** may be contained in various envelope types (e.g. X.509 attribute  
437 certificates, SAML attribute assertions, etc.). Therefore, there is a need for a canonical form of the  
438 request and response handled by an XACML **PDP**. This canonical form is called the XACML **context**. Its  
439 syntax is defined in XML schema.

440 Naturally, XACML-conformant **PEPs** may issue requests and receive responses in the form of an XACML  
441 **context**. But, where this situation does not exist, an intermediate step is required to convert between the  
442 request/response format understood by the **PEP** and the XACML **context** format understood by the **PDP**.

443 The benefit of this approach is that **policies** may be written and analyzed independently of the specific  
444 environment in which they are to be enforced.

445 In the case where the native request/response format is specified in XML Schema (e.g. a SAML-  
446 conformant **PEP**), the transformation between the native format and the XACML **context** may be  
447 specified in the form of an Extensible Stylesheet Language Transformation [XSLT].

448 Similarly, in the case where the **resource** to which **access** is requested is an XML document, the  
449 **resource** itself may be included in, or referenced by, the request **context**. Then, through the use of  
450 XPath expressions [XPath] in the **policy**, values in the **resource** may be included in the **policy**  
451 evaluation.

## 452 2.12 Actions performed in conjunction with enforcement

453 In many applications, **policies** specify actions that MUST be performed, either instead of, or in addition  
454 to, actions that MAY be performed. This idea was described by Sloman [Sloman94]. XACML provides  
455 facilities to specify actions that MUST be performed in conjunction with **policy** evaluation in the  
456 <Obligations> element. This idea was described as a provisional action by Kudo [Kudo00]. There  
457 are no standard definitions for these actions in version 3.0 of XACML. Therefore, bilateral agreement  
458 between a **PAP** and the **PEP** that will enforce its **policies** is required for correct interpretation. **PEPs** that  
459 conform to v3.0 of XACML are required to deny **access** unless they understand and can discharge all of  
460 the <Obligations> elements associated with the **applicable policy**. <Obligations> elements are  
461 returned to the **PEP** for enforcement.

## 462 2.13 Supplemental information about a decision

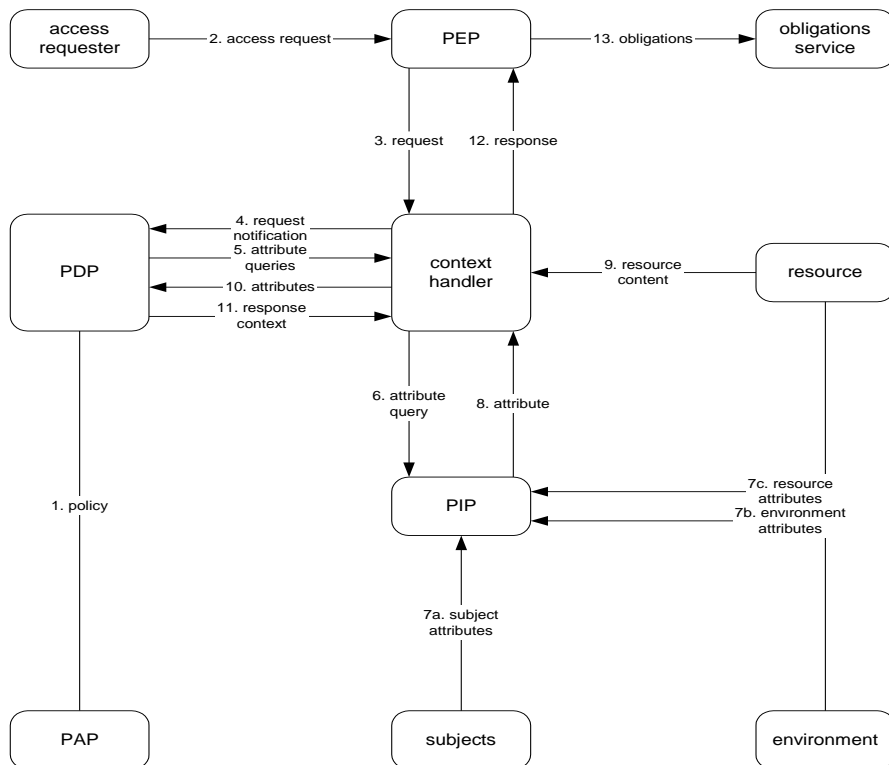
463 In some applications it is helpful to specify supplemental information about a decision. XACML provides  
464 facilities to specify supplemental information about a decision with the <Advice> element. Such **advice**  
465 may be safely ignored by the **PEP**.

466 **3 Models (non-normative)**

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

468 **3.1 Data-flow model**

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



470  
471 *Figure 1 - Data-flow diagram*

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

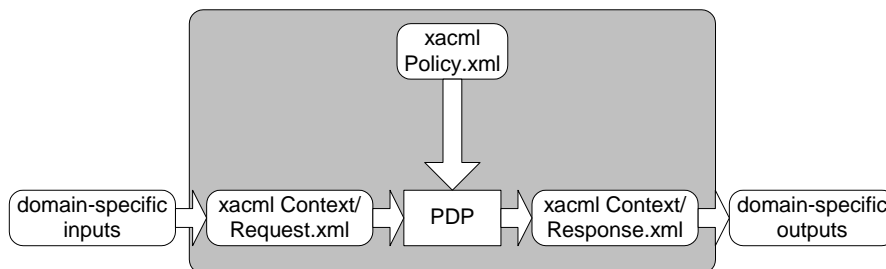
478 The model operates by the following steps.

- 479 1. **PAPs** write **policies** and **policy sets** and make them available to the **PDP**. These **policies** or  
480 **policy sets** represent the complete **policy** for a specified **target**.
- 481 2. The **access** requester sends a request for **access** to the **PEP**.

- 482 3. The **PEP** sends the request for **access** to the **context handler** in its native request format,  
483 optionally including **attributes** of the **subjects**, **resource**, **action**, **environment** and other  
484 categories.
- 485 4. The **context handler** constructs an XACML request **context**, optionally adds attributes, and  
486 sends it to the **PDP**.
- 487 5. The **PDP** requests any additional **subject**, **resource**, **action**, **environment** and other categories  
488 (not shown) **attributes** from the **context handler**.
- 489 6. The **context handler** requests the **attributes** from a **PIP**.
- 490 7. The **PIP** obtains the requested **attributes**.
- 491 8. The **PIP** returns the requested **attributes** to the **context handler**.
- 492 9. Optionally, the **context handler** includes the **resource** in the **context**.
- 493 10. The **context handler** sends the requested **attributes** and (optionally) the **resource** to the **PDP**.  
494 The **PDP** evaluates the **policy**.
- 495 11. The **PDP** returns the response **context** (including the **authorization decision**) to the **context**  
496 **handler**.
- 497 12. The **context handler** translates the response **context** to the native response format of the **PEP**.  
498 The **context handler** returns the response to the **PEP**.
- 499 13. The **PEP** fulfills the **obligations**.
- 500 14. (Not shown) If **access** is permitted, then the **PEP** permits **access** to the **resource**; otherwise, it  
501 denies **access**.

### 502 3.2 XACML context

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



514  
515 *Figure 2 - XACML context*

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

518 Typical categories of **attributes** in the **context** are the **subject**, **resource**, **action** and **environment**, but  
519 users may define their own categories as needed. See appendix B.2 for suggested **attribute** categories.

520 See Section 7.3.5 for a more detailed discussion of the request **context**.

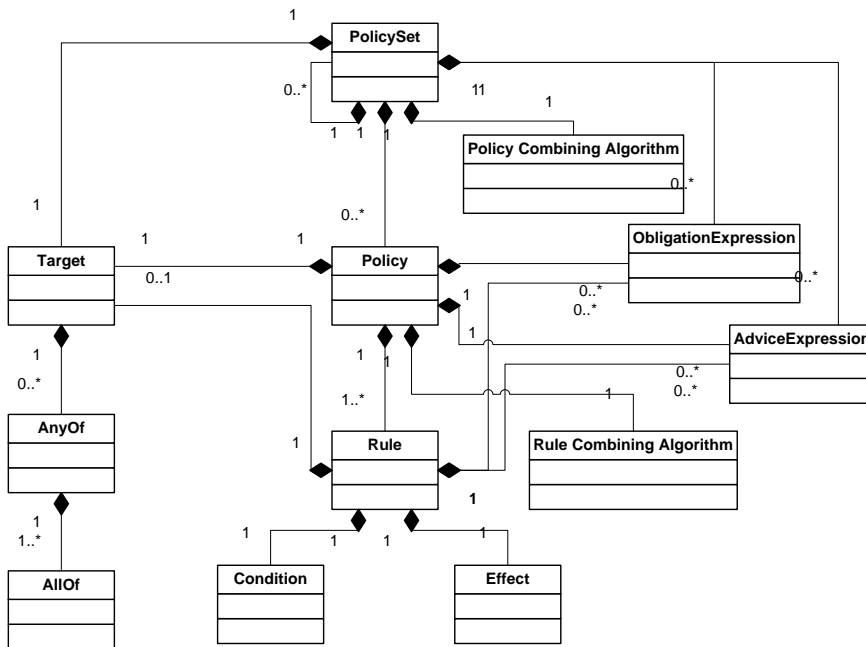
521 **3.3 Policy language model**

522 The *policy* language model is shown in Figure 3. The main components of the model are:

- 523 • **Rule**;
- 524 • **Policy**; and
- 525 • **Policy set**.

526 These are described in the following sub-sections.

527



528  
529 Figure 3 - Policy language model

530 **3.3.1 Rule**

531 A *rule* is the most elementary unit of *policy*. It may exist in isolation only within one of the major actors of  
532 the XACML domain. In order to exchange *rules* between major actors, they must be encapsulated in a  
533 *policy*. A *rule* can be evaluated on the basis of its contents. The main components of a *rule* are:

- 534 • a **target**;
- 535 • an **effect**;
- 536 • a **condition**;
- 537 • **obligation** expressions, and
- 538 • **advice** expressions

539 These are discussed in the following sub-sections.

### 540 3.3.1.1 Rule target

541 The **target** defines the set of requests to which the **rule** is intended to apply in the form of a logical  
542 expression on **attributes** in the request. The <Condition> element may further refine the applicability  
543 established by the **target**. If the **rule** is intended to apply to all entities of a particular data-type, then the  
544 corresponding entity is omitted from the **target**. An XACML **PDP** verifies that the matches defined by the  
545 **target** are satisfied by the **attributes** in the request **context**.

546 The <Target> element may be absent from a <Rule>. In this case, the **target** of the <Rule> is the  
547 same as that of the parent <Policy> element.

548 Certain **subject** name-forms, **resource** name-forms and certain types of **resource** are internally  
549 structured. For instance, the X.500 directory name-form and RFC 822 name-form are structured **subject**  
550 name-forms, whereas an account number commonly has no discernible structure. UNIX file-system path-  
551 names and URIs are examples of structured **resource** name-forms. An XML document is an example of  
552 a structured **resource**.

553 Generally, the name of a node (other than a leaf node) in a structured name-form is also a legal instance  
554 of the name-form. So, for instance, the RFC822 name "med.example.com" is a legal RFC822 name  
555 identifying the set of mail addresses hosted by the med.example.com mail server. The XPath value  
556 md:record/md:patient/ is a legal XPath value identifying a node-set in an XML document.

557 The question arises: how should a name that identifies a set of **subjects** or **resources** be interpreted by  
558 the **PDP**, whether it appears in a **policy** or a request **context**? Are they intended to represent just the  
559 node explicitly identified by the name, or are they intended to represent the entire sub-tree subordinate to  
560 that node?

561 In the case of **subjects**, there is no real entity that corresponds to such a node. So, names of this type  
562 always refer to the set of **subjects** subordinate in the name structure to the identified node.

563 Consequently, non-leaf **subject** names should not be used in equality functions, only in match functions,  
564 such as "urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match" not  
565 "urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal" (see Appendix 10.2.9).

### 566 3.3.1.2 Effect

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

### 569 3.3.1.3 Condition

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

### 572 3.3.1.4 Obligation expressions

573 **Obligation** expressions may be added by the writer of the **rule**.

574 When a **PDP** evaluates a **rule** containing **obligation** expressions, it evaluates the **obligation** expressions  
575 into **obligations** and returns certain of those **obligations** to the **PEP** in the response **context**. Section  
576 7.18 explains which **obligations** are to be returned.

### 577 3.3.1.5 Advice

578 **Advice** expressions may be added by the writer of the **rule**.

579 When a **PDP** evaluates a **rule** containing **advice** expressions, it evaluates the **advice** expressions into  
580 **advice** and returns certain of those **advice** to the **PEP** in the response **context**. Section 7.18 explains  
581 which **advice** are to be returned. In contrast to **obligations**, **advice** may be safely ignored by the **PEP**.

## 582 3.3.2 Policy

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

- 585 • a **target**;
  - 586 • a **rule-combining algorithm**-identifier;
  - 587 • a set of **rules**;
  - 588 • **obligation** expressions and
  - 589 • **advice** expressions
- 590 **Rules** are described above. The remaining components are described in the following sub-sections.

### 591 3.3.2.1 Policy target

592 An XACML <PolicySet>, <Policy> or <Rule> element contains a <Target> element that specifies  
593 the set of requests to which it applies. The <Target> of a <PolicySet> or <Policy> may be declared  
594 by the writer of the <PolicySet> or <Policy>, or it may be calculated from the <Target> elements of  
595 the <PolicySet>, <Policy> and <Rule> elements that it contains.

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

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

### 611 3.3.2.2 Rule-combining algorithm

612 The **rule-combining algorithm** specifies the procedure by which the results of evaluating the component  
613 **rules** are combined when evaluating the **policy**, i.e. the **decision** value placed in the response **context**  
614 by the **PDP** is the value of the **policy**, as defined by the **rule-combining algorithm**. A **policy** may have  
615 combining parameters that affect the operation of the **rule-combining algorithm**.

616 See Appendix Appendix C for definitions of the normative **rule-combining algorithms**.

### 617 3.3.2.3 Obligation expressions

618 **Obligation** expressions may be added by the writer of the **policy**.

619 When a **PDP** evaluates a **policy** containing **obligation** expressions, it evaluates the **obligation**  
620 expressions into **obligations** and returns certain of those **obligations** to the **PEP** in the response  
621 **context**. Section 7.18 explains which **obligations** are to be returned.

### 622 3.3.2.4 Advice

623 **Advice** expressions may be added by the writer of the **policy**.

624 When a **PDP** evaluates a **policy** containing **advice** expressions, it evaluates the **advice** expressions into  
625 **advice** and returns certain of those **advice** to the **PEP** in the response **context**. Section 7.18 explains  
626 which **advice** are to be returned. In contrast to **obligations**, **advice** may be safely ignored by the **PEP**.

### 627 3.3.3 Policy set

628 A **policy set** comprises four main components:

- 629 • a **target**;
- 630 • a **policy-combining algorithm**-identifier
- 631 • a set of **policies**;
- 632 • **obligation** expressions, and
- 633 • **advice** expressions

634 The **target** and **policy** components are described above. The other components are described in the  
635 following sub-sections.

#### 636 3.3.3.1 Policy-combining algorithm

637 The **policy-combining algorithm** specifies the procedure by which the results of evaluating the  
638 component **policies** are combined when evaluating the **policy set**, i.e. the `Decision` value placed in the  
639 response **context** by the **PDP** is the result of evaluating the **policy set**, as defined by the **policy-**  
640 **combining algorithm**. A **policy set** may have combining parameters that affect the operation of the  
641 **policy-combining algorithm**.

642 See Appendix Appendix C for definitions of the normative **policy-combining algorithms**.

#### 643 3.3.3.2 Obligation expressions

644 The writer of a **policy set** may add **obligation** expressions to the **policy set**, in addition to those  
645 contained in the component **rules**, **policies** and **policy sets**.

646 When a **PDP** evaluates a **policy set** containing **obligations** expressions, it evaluates the **obligation**  
647 expressions into **obligations** and returns certain of those **obligations** to the **PEP** in its response **context**.  
648 Section 7.18 explains which **obligations** are to be returned.

#### 649 3.3.3.3 Advice expressions

650 **Advice** expressions may be added by the writer of the **policy set**.

651 When a **PDP** evaluates a **policy set** containing **advice** expressions, it evaluates the **advice** expressions  
652 into **advice** and returns certain of those **advice** to the **PEP** in the response **context**. Section 7.18  
653 explains which **advice** are to be returned. In contrast to **obligations**, **advice** may be safely ignored by  
654 the **PEP**.



## 655 4 Examples (non-normative)

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

### 660 4.1 Example one

#### 661 4.1.1 Example policy

662 Assume that a corporation named Medi Corp (identified by its domain name: med.example.com) has an  
663 **access control policy** that states, in English:

664 *Any user with an e-mail name in the "med.example.com" namespace is allowed to perform any **action** on  
665 any resource.*

666 An XACML **policy** consists of header information, an optional text description of the **policy**, a **target**, one  
667 or more **rules** and an optional set of **obligation** expressions.

```
668 [a1] <?xml version="1.0" encoding="UTF-8"?>
669 [a2] <Policy
670 [a3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
671 [a4]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
672 [a5]   xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
673 [a6]   http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
674 [a7]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:SimplePolicy1"
675 [a8]   Version="1.0"
676 [a9]   RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">
677 [a10] <Description>
678 [a11]   Medi Corp access control policy
679 [a12] </Description>
680 [a13] <Target/>
681 [a14] <Rule
682 [a15]   RuleId="urn:oasis:names:tc:xacml:3.0:example:SimpleRule1"
683 [a16]   Effect="Permit">
684 [a17] <Description>
685 [a18]   Any subject with an e-mail name in the med.example.com domain
686 [a19]   can perform any action on any resource.
687 [a20] </Description>
688 [a21] <Target>
689 [a22]   <AnyOf>
690 [a23]     <AllOf>
691 [a24]       <Match
692 [a25]         MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
693 [a26]           <AttributeValue
694 [a27]             DataType="http://www.w3.org/2001/XMLSchema#string"
695 [a28]             >med.example.com</AttributeValue>
696 [a29]           <AttributeDesignator
697 [a30]             MustBePresent="false"
698 [a31]             Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
699 [a32]             subject"
700 [a32]             AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
701 [a33]             DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
702 [a34]           </Match>
703 [a35]         </AllOf>
704 [a36]       </AnyOf>
705 [a37]     </Target>
706 [a38]   </Rule>
707 [a39] </Policy>
```

708 [a1] is a standard XML document tag indicating which version of XML is being used and what the  
709 character encoding is.

710 [a2] introduces the XACML **Policy** itself.

711 [a3] - [a4] are XML namespace declarations.  
712 [a3] gives a URN for the XACML *policies* schema.  
713 [a7] assigns a name to this *policy* instance. The name of a *policy* has to be unique for a given *PDP* so  
714 that there is no ambiguity if one *policy* is referenced from another *policy*. The version attribute specifies  
715 the version of this policy is "1.0".  
716 [a9] specifies the algorithm that will be used to resolve the results of the various *rules* that may be in the  
717 *policy*. The deny-overrides *rule-combining algorithm* specified here says that, if any *rule* evaluates to  
718 "Deny", then the *policy* must return "Deny". If all *rules* evaluate to "Permit", then the *policy* must return  
719 "Permit". The *rule-combining algorithm*, which is fully described in Appendix Appendix C, also says  
720 what to do if an error were to occur when evaluating any *rule*, and what to do with *rules* that do not apply  
721 to a particular *decision request*.  
722 [a10] - [a12] provide a text description of the *policy*. This description is optional.  
723 [a13] describes the *decision requests* to which this *policy* applies. If the *attributes* in a *decision*  
724 *request* do not match the values specified in the *policy target*, then the remainder of the *policy* does not  
725 need to be evaluated. This *target* section is useful for creating an index to a set of *policies*. In this  
726 simple example, the *target* section says the *policy* is applicable to any *decision request*.  
727 [a14] introduces the one and only *rule* in this simple *policy*.  
728 [a15] specifies the identifier for this *rule*. Just as for a *policy*, each *rule* must have a unique identifier (at  
729 least unique for any *PDP* that will be using the *policy*).  
730 [a16] says what *effect* this *rule* has if the *rule* evaluates to "True". *Rules* can have an *effect* of either  
731 "Permit" or "Deny". In this case, if the *rule* is satisfied, it will evaluate to "Permit", meaning that, as far as  
732 this one *rule* is concerned, the requested *access* should be permitted. If a *rule* evaluates to "False",  
733 then it returns a result of "NotApplicable". If an error occurs when evaluating the *rule*, then the *rule*  
734 returns a result of "Indeterminate". As mentioned above, the *rule-combining algorithm* for the *policy*  
735 specifies how various *rule* values are combined into a single *policy* value.  
736 [a17] - [a20] provide a text description of this *rule*. This description is optional.  
737 [a21] introduces the *target* of the *rule*. As described above for the *target* of a *policy*, the *target* of a *rule*  
738 describes the *decision requests* to which this *rule* applies. If the *attributes* in a *decision request* do  
739 not match the values specified in the *rule target*, then the remainder of the *rule* does not need to be  
740 evaluated, and a value of "NotApplicable" is returned to the *rule* evaluation.  
741 The *rule target* is similar to the *target* of the *policy* itself, but with one important difference. [a22] - [a36]  
742 spells out a specific value that the *subject* in the *decision request* must match. The <Match> element  
743 specifies a *matching function* in the *MatchId* attribute, a literal value of "med.example.com" and a pointer  
744 to a specific *subject attribute* in the request *context* by means of the <AttributeDesignator>  
745 element with an *attribute* category which specifies the *access subject*. The matching function will be  
746 used to compare the literal value with the value of the *subject attribute*. Only if the match returns "True"  
747 will this *rule* apply to a particular *decision request*. If the match returns "False", then this *rule* will return  
748 a value of "NotApplicable".  
749 [a38] closes the *rule*. In this *rule*, all the work is done in the <Target> element. In more complex *rules*,  
750 the <Target> may have been followed by a <Condition> element (which could also be a set of  
751 *conditions* to be ANDed or ORed together).  
752 [a39] closes the *policy*. As mentioned above, this *policy* has only one *rule*, but more complex *policies*  
753 may have any number of *rules*.

#### 754 4.1.2 Example request context

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

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

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

```

761 [b1] <?xml version="1.0" encoding="UTF-8"?>
762 [b2] <Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
763 [b3] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
764 [b4] xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
765 [b5] http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
766 [b5] ReturnPolicyIdList="false">
767 [b6] <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
768 [b6] subject">
769 [b7] <Attribute IncludeInResult="false"
770 [b8] <AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
771 [b9] <AttributeValue
772 [b10] DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"
773 [b11] >bs@simpsons.com</AttributeValue>
774 [b12] </Attribute>
775 [b13] </Attributes>
776 [b14] <Attributes
777 [b15] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
778 [b16] <Attribute IncludeInResult="false"
779 [b17] <AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id">
780 [b18] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema:anyURI"
781 [b19] >file://example/med/record/patient/BartSimpson</AttributeValue>
782 [b20] </Attribute>
783 [b21] </Attributes>
784 [b22] <Attributes
785 [b23] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
786 [b24] <Attribute IncludeInResult="false"
787 [b25] <AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
788 [b26] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema:string"
789 [b27] >read</AttributeValue>
790 [b28] </Attribute>
791 [b29] </Attributes>
792 [b30] </Request>

```

793 [b1] - [b2] contain the header information for the request **context**, and are used the same way as the  
794 header for the **policy** explained above.

795 The first <Attributes> element contains **attributes** of the entity making the **access** request. There  
796 can be multiple **subjects** in the form of additional <Attributes> elements with different categories, and  
797 each **subject** can have multiple **attributes**. In this case, in [b6] - [b13], there is only one **subject**, and the  
798 **subject** has only one **attribute**: the **subject's** identity, expressed as an e-mail name, is  
799 "bs@simpsons.com".

800 The second <Attributes> element contains **attributes** of the **resource** to which the **subject** (or  
801 **subjects**) has requested **access**. Lines [b14] - [b21] contain the one **attribute** of the **resource** to which  
802 Bart Simpson has requested **access**: the **resource** identified by its file URI, which is  
803 "file://medico/record/patient/BartSimpson".

804 The third <Attributes> element contains **attributes** of the **action** that the **subject** (or **subjects**)  
805 wishes to take on the **resource**. [b22] - [b29] describe the identity of the **action** Bart Simpson wishes to  
806 take, which is "read".

807 [b30] closes the request **context**. A more complex request **context** may have contained some **attributes**  
808 not associated with the **subject**, the **resource** or the **action**. Environment would be an example of such  
809 an attribute category. These would have been placed in additional <Attributes> elements. Examples  
810 of such **attributes** are **attributes** describing the **environment** or some application specific category of  
811 **attributes**.

812 The **PDP** processing this request **context** locates the **policy** in its **policy** repository. It compares the  
813 **attributes** in the request **context** with the **policy target**. Since the **policy target** is empty, the **policy**  
814 matches this **context**.

815 The **PDP** now compares the **attributes** in the request **context** with the **target** of the one **rule** in this  
816 **policy**. The requested **resource** matches the <Target> element and the requested **action** matches the  
817 <Target> element, but the requesting **subject-id attribute** does not match "med.example.com".

### 818 4.1.3 Example response context

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

```
822 [c1] <?xml version="1.0" encoding="UTF-8"?>  
823 [c2] <Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"  
824     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
825     xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17  
826     http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd">  
827 [c3]   <Result>  
828 [c4]     <Decision>NotApplicable</Decision>  
829 [c5]   </Result>  
830 [c6] </Response>
```

831 [c1] - [c2] contain the same sort of header information for the response as was described above for a  
832 **policy**.

833 The <Result> element in lines [c3] - [c5] contains the result of evaluating the **decision request** against  
834 the **policy**. In this case, the result is "NotApplicable". A **policy** can return "Permit", "Deny",  
835 "NotApplicable" or "Indeterminate". Therefore, the **PEP** is required to deny **access**.

836 [c6] closes the response **context**.

## 837 4.2 Example two

838 This section contains an example XML document, an example request **context** and example XACML  
839 **rules**. The XML document is a medical record. Four separate **rules** are defined. These illustrate a **rule-**  
840 **combining algorithm**, **conditions** and **obligation** expressions.

### 841 4.2.1 Example medical record instance

842 The following is an instance of a medical record to which the example XACML **rules** can be applied. The  
843 <record> schema is defined in the registered namespace administered by Medi Corp.

```
844 [d1] <?xml version="1.0" encoding="UTF-8"?>  
845 [d2] <record xmlns="urn:example:med:schemas:record"  
846 [d3]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
847 [d4]   <patient>  
848 [d5]     <patientName>  
849 [d6]       <first>Bartholomew</first>  
850 [d7]       <last>Simpson</last>  
851 [d8]     </patientName>  
852 [d9]     <patientContact>  
853 [d10]       <street>27 Shelbyville Road</street>  
854 [d11]       <city>Springfield</city>  
855 [d12]       <state>MA</state>  
856 [d13]       <zip>12345</zip>  
857 [d14]       <phone>555.123.4567</phone>  
858 [d15]       <fax/>  
859 [d16]       <email/>  
860 [d17]     </patientContact>  
861 [d18]     <patientDoB>1992-03-21</patientDoB>  
862 [d19]     <patientGender>male</patientGender>  
863 [d20]     <patient-number>555555</patient-number>  
864 [d21]   </patient>  
865 [d22]   <parentGuardian>  
866 [d23]     <parentGuardianId>HS001</parentGuardianId>  
867 [d24]     <parentGuardianName>  
868 [d25]       <first>Homer</first>  
869 [d26]       <last>Simpson</last>  
870 [d27]     </parentGuardianName>  
871 [d28]     <parentGuardianContact>  
872 [d29]       <street>27 Shelbyville Road</street>  
873 [d30]       <city>Springfield</city>  
874 [d31]       <state>MA</state>  
875 [d32]       <zip>12345</zip>  
876 [d33]       <phone>555.123.4567</phone>  
877 [d34]       <fax/>
```

```

878 [d35] <email>homers@aol.com</email>
879 [d36] </parentGuardianContact>
880 [d37] </parentGuardian>
881 [d38] <primaryCarePhysician>
882 [d39] <physicianName>
883 [d40] <first>Julius</first>
884 [d41] <last>Hibbert</last>
885 [d42] </physicianName>
886 [d43] <physicianContact>
887 [d44] <street>1 First St</street>
888 [d45] <city>Springfield</city>
889 [d46] <state>MA</state>
890 [d47] <zip>12345</zip>
891 [d48] <phone>555.123.9012</phone>
892 [d49] <fax>555.123.9013</fax>
893 [d50] <email/>
894 [d51] </physicianContact>
895 [d52] <registrationID>ABC123</registrationID>
896 [d53] </primaryCarePhysician>
897 [d54] <insurer>
898 [d55] <name>Blue Cross</name>
899 [d56] <street>1234 Main St</street>
900 [d57] <city>Springfield</city>
901 [d58] <state>MA</state>
902 [d59] <zip>12345</zip>
903 [d60] <phone>555.123.5678</phone>
904 [d61] <fax>555.123.5679</fax>
905 [d62] <email/>
906 [d63] </insurer>
907 [d64] <medical>
908 [d65] <treatment>
909 [d66] <drug>
910 [d67] <name>methylphenidate hydrochloride</name>
911 [d68] <dailyDosage>30mgs</dailyDosage>
912 [d69] <startDate>1999-01-12</startDate>
913 [d70] </drug>
914 [d71] <comment>
915 [d72] patient exhibits side-effects of skin coloration and carpal degeneration
916 [d73] </comment>
917 [d74] </treatment>
918 [d75] <result>
919 [d76] <test>blood pressure</test>
920 [d77] <value>120/80</value>
921 [d78] <date>2001-06-09</date>
922 [d79] <performedBy>Nurse Betty</performedBy>
923 [d80] </result>
924 [d81] </medical>
925 [d82] </record>

```

## 926 4.2.2 Example request context

927 The following example illustrates a request *context* to which the example *rules* may be applicable. It  
928 represents a request by the physician Julius Hibbert to read the patient date of birth in the record of  
929 Bartholomew Simpson.

```

930 [e1] <?xml version="1.0" encoding="UTF-8"?>
931 [e2] <Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
932 [e3] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
933 [e4] xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
935 [e5] ReturnPolicyIdList="false">
936 [e6] <Attributes
937 [e7] Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
938 [e8] <Attribute IncludeInResult="false"
939 [e9] AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
940 [e10] Issuer="med.example.com">
941 [e11] <AttributeValue
942 [e12] DataType="http://www.w3.org/2001/XMLSchema#string">CN=Julius
Hibbert</AttributeValue>
944 [e13] </Attribute>
945 [e14] <Attribute IncludeInResult="false"
946 [e15] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"

```

```

947 [e16] Issuer="med.example.com">
948 [e17] <AttributeValue
949 [e18]   DataType="http://www.w3.org/2001/XMLSchema#string"
950 [e19]   >physician</AttributeValue>
951 [e20] </Attribute>
952 [e21] <Attribute IncludeInResult="false"
953 [e22]   AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:physician-id"
954 [e23]   Issuer="med.example.com">
955 [e24]   <AttributeValue
956 [e25]     DataType="http://www.w3.org/2001/XMLSchema#string">jh1234</AttributeValue>
957 [e26]   </Attribute>
958 [e27] </Attributes>
959 [e28] <Attributes
960 [e29]   Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
961 [e30]   <Content>
962 [e31]     <md:record xmlns:md="urn:example:med:schemas:record"
963 [e32]       xsi:schemaLocation="urn:example:med:schemas:record
964 [e33]         http://www.med.example.com/schemas/record.xsd">
965 [e34]       <md:patient>
966 [e35]         <md:patientDoB>1992-03-21</md:patientDoB>
967 [e36]         <md:patient-number>555555</md:patient-number>
968 [e37]         <md:patientContact>
969 [e38]           <md:email>b.simpson@example.com</md:email>
970 [e39]         </md:patientContact>
971 [e40]       </md:patient>
972 [e41]     </md:record>
973 [e42]   </Content>
974 [e43]   <Attribute IncludeInResult="false"
975 [e44]     AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector" >
976 [e45]   <AttributeValue
977 [e46]     XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
978 [e47]     DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
979 [e48]     >md:record/md:patient/md:patientDoB</AttributeValue>
980 [e49]   </Attribute>
981 [e50]   <Attribute IncludeInResult="false"
982 [e51]     AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace" >
983 [e52]   <AttributeValue
984 [e53]     DataType="http://www.w3.org/2001/XMLSchema#anyURI"
985 [e54]     >urn:example:med:schemas:record</AttributeValue>
986 [e55]   </Attribute>
987 [e56] </Attributes>
988 [e57] <Attributes
989 [e58]   Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
990 [e59]   <Attribute IncludeInResult="false"
991 [e60]     AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" >
992 [e61]   <AttributeValue
993 [e62]     DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
994 [e63]   </Attribute>
995 [e64] </Attributes>
996 [e65] <Attributes
997 [e66]   Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment">
998 [e67]   <Attribute IncludeInResult="false"
999 [e68]     AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date" >
1000 [e69]   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#date"
1001 [e70]     >2010-01-11</AttributeValue>
1002 [e71]   </Attribute>
1003 [e72] </Attributes>
1004 [e73] </Request>

```

1005 [e2] - [e4] Standard namespace declarations.

1006 [e6] - [e27] **Access subject attributes** are placed in the urn:oasis:names:tc:xacml:1.0:subject-  
1007 category:access-subject **attribute** category of the <Request> element. Each **attribute** consists of the  
1008 **attribute** meta-data and the **attribute** value. There is only one **subject** involved in this request. This  
1009 value of the **attribute** category denotes the identity for which the request was issued.

1010 [e8] - [e13] **Subject subject-id attribute**.

1011 [e14] - [e20] **Subject role attribute**.

1012 [e21] - [e26] **Subject physician-id attribute**.

1013 [e28] - [e56] **Resource attributes** are placed in the urn:oasis:names:tc:xacml:3.0:attribute-  
 1014 category:resource **attribute** category of the <Request> element. Each **attribute** consists of **attribute**  
 1015 meta-data and an **attribute** value.

1016 [e30] - [e42] **Resource** content. The XML **resource** instance, **access** to all or part of which may be  
 1017 requested, is placed here.

1018 [e43] - [e49] The identifier of the **Resource** instance for which **access** is requested, which is an XPath  
 1019 expression into the <Content> element that selects the data to be accessed.

1020 [e57] - [e64] **Action attributes** are placed in the urn:oasis:names:tc:xacml:3.0:attribute-category:action  
 1021 **attribute** category of the <Request> element.

1022 [e59] - [e63] **Action** identifier.

### 1023 4.2.3 Example plain-language rules

1024 The following plain-language **rules** are to be enforced:

- 1025 Rule 1: A person, identified by his or her patient number, may read any record for which he or she is  
 1026 the designated patient.
- 1027 Rule 2: A person may read any record for which he or she is the designated parent or guardian, and  
 1028 for which the patient is under 16 years of age.
- 1029 Rule 3: A physician may write to any medical element for which he or she is the designated primary  
 1030 care physician, provided an email is sent to the patient.
- 1031 Rule 4: An administrator shall not be permitted to read or write to medical elements of a patient  
 1032 record.

1033 These **rules** may be written by different **PAPs** operating independently, or by a single **PAP**.

### 1034 4.2.4 Example XACML rule instances

#### 1035 4.2.4.1 Rule 1

1036 **Rule 1** illustrates a simple **rule** with a single <Condition> element. It also illustrates the use of the  
 1037 <VariableDefinition> element to define a function that may be used throughout the **policy**. The  
 1038 following XACML <Rule> instance expresses **Rule 1**:

```

1039 [f1] <?xml version="1.0" encoding="UTF-8"?>
1040 [f2] <Policy
1041 [f3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1042 [f4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1043 [f5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1044 [f6]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1045 [f7]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:1"
1046 [f8]   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1047   algorithm:deny-overrides"
1048 [f9]   Version="1.0">
1049 [f10] <PolicyDefaults>
1050 [f11]   <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1051 [f12] </PolicyDefaults>
1052 [f13] <Target/>
1053 [f14] <VariableDefinition VariableId="17590034">
1054 [f15]   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1055 [f16]     <Apply
1056 [f17]       FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1057 [f18]         <AttributeDesignator
1058 [f19]           MustBePresent="false"
1059 [f20]           Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
1060   subject"
1061 [f21]           AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:patient-
1062   number"
1063 [f22]           DataType="http://www.w3.org/2001/XMLSchema#string"/>
1064 [f23]         </Apply>
1065 [f24]       </Apply>
1066 [f25]     <Apply
    
```

```

1067 [f26] <AttributeSelector
1068 [f27]     MustBePresent="false"
1069 [f28]     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1070 [f29]     Path="md:record/md:patient/md:patient-number/text()"
1071 [f30]     DataType="http://www.w3.org/2001/XMLSchema#string"/>
1072 [f31] </Apply>
1073 [f32] </Apply>
1074 [f33] </VariableDefinition>
1075 [f34] <Rule
1076 [f35]     RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"
1077 [f36]     Effect="Permit">
1078 [f37]     <Description>
1079 [f38]         A person may read any medical record in the
1080 [f39]         http://www.med.example.com/schemas/record.xsd namespace
1081 [f40]         for which he or she is the designated patient
1082 [f41]     </Description>
1083 [f42]     <Target>
1084 [f43]         <AnyOf>
1085 [f44]             <AllOf>
1086 [f45]                 <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1087 [f46]                     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1088 [f47]                         >urn:example:med:schemas:record</AttributeValue>
1089 [f48]                     <AttributeDesignator
1090 [f49]                         MustBePresent="false"
1091 [f50]                         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1092 [f51]                         AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1093 [f52]                         DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1094 [f53]                 </Match>
1095 [f54]                 <Match
1096 [f55]                     MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1097 [f56]                     <AttributeValue
1098 [f57]                         DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1099 [f58]                     XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1100 [f59]                         >md:record</AttributeValue>
1101 [f60]                     <AttributeDesignator
1102 [f61]                         MustBePresent="false"
1103 [f62]                         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1104 [f63]                         AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1105 [f64]                         DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1106 [f65]                     </Match>
1107 [f66]                 </AllOf>
1108 [f67]             </AnyOf>
1109 [f68]         </AnyOf>
1110 [f69]     <AllOf>
1111 [f70]         <Match
1112 [f71]             MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1113 [f72]                 <AttributeValue
1114 [f73]                     DataType="http://www.w3.org/2001/XMLSchema#string"
1115 [f74]                     >read</AttributeValue>
1116 [f75]                 <AttributeDesignator
1117 [f76]                     MustBePresent="false"
1118 [f77]                     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1119 [f78]                     AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1120 [f79]                     DataType="http://www.w3.org/2001/XMLSchema#string"/>
1121 [f80]                 </Match>
1122 [f81]             </AllOf>
1123 [f82]         </AnyOf>
1124 [f83]     </Target>
1125 [f84]     <Condition>
1126 [f85]         <VariableReference VariableId="17590034"/>
1127 [f86]     </Condition>
1128 [f87] </Rule>
1129 [f88] </Policy>

```

1130 [f3] - [f6] XML namespace declarations.

1131 [f11] XPath expressions in the *policy* are to be interpreted according to the 1.0 version of the XPath specification.

1133 [f14] - [f33] A <VariableDefinition> element. It defines a function that evaluates the truth of the statement: the patient-number *subject attribute* is equal to the patient-number in the *resource*.



1135 [f15] The `FunctionId` attribute names the function to be used for comparison. In this case, comparison  
1136 is done with the "urn:oasis:names:tc:xacml:1.0:function:string-equal" function; this function takes two  
1137 arguments of type "http://www.w3.org/2001/XMLSchema#string".

1138 [f17] The first argument of the variable definition is a function specified by the `FunctionId` attribute.  
1139 Since urn:oasis:names:tc:xacml:1.0:function:string-equal takes arguments of type  
1140 "http://www.w3.org/2001/XMLSchema#string" and `AttributeDesignator` selects a **bag** of type  
1141 "http://www.w3.org/2001/XMLSchema#string", "urn:oasis:names:tc:xacml:1.0:function:string-one-and-  
1142 only" is used. This function guarantees that its argument evaluates to a **bag** containing exactly one  
1143 value.

1144 [f18] The `AttributeDesignator` selects a **bag** of values for the patient-number **subject attribute** in  
1145 the request **context**.

1146 [f25] The second argument of the variable definition is a function specified by the `FunctionId` attribute.  
1147 Since "urn:oasis:names:tc:xacml:1.0:function:string-equal" takes arguments of type  
1148 "http://www.w3.org/2001/XMLSchema#string" and the `AttributeSelector` selects a **bag** of type  
1149 "http://www.w3.org/2001/XMLSchema#string", "urn:oasis:names:tc:xacml:1.0:function:string-one-and-  
1150 only" is used. This function guarantees that its argument evaluates to a **bag** containing exactly one  
1151 value.

1152 [f26] The `<AttributeSelector>` element selects a **bag** of values from the **resource** content using a  
1153 free-form XPath expression. In this case, it selects the value of the patient-number in the **resource**.  
1154 Note that the namespace prefixes in the XPath expression are resolved with the standard XML  
1155 namespace declarations.

1156 [f35] **Rule** identifier.

1157 [f36] **Rule effect** declaration. When a **rule** evaluates to 'True' it emits the value of the `Effect` attribute.  
1158 This value is then combined with the `Effect` values of other **rules** according to the **rule-combining**  
1159 **algorithm**.

1160 [f37] - [f41] Free form description of the **rule**.

1161 [f42] - [f83] A **rule target** defines a set of **decision requests** that the **rule** is intended to evaluate.

1162 [f43] - [f67] The `<AnyOf>` element contains a **disjunctive sequence** of `<AllOf>` elements. In this  
1163 example, there is just one.

1164 [f44] - [f66] The `<AllOf>` element encloses the **conjunctive sequence** of `Match` elements. In this  
1165 example, there are two.

1166 [f45] - [f53] The first `<Match>` element compares its first and second child elements according to the  
1167 matching function. A match is positive if the value of the first argument matches any of the values  
1168 selected by the second argument. This match compares the **target** namespace of the requested  
1169 document with the value of "urn:example:med:schemas:record".

1170 [f45] The `MatchId` attribute names the matching function.

1171 [f46] - [f47] Literal **attribute** value to match.

1172 [f48] - [f52] The `<AttributeDesignator>` element selects the **target** namespace from the **resource**  
1173 contained in the request **context**. The **attribute** name is specified by the `AttributeId`.

1174 [f54] - [f65] The second `<Match>` element. This match compares the results of two XPath expressions  
1175 applied to the `<Content>` element of the **resource** category. The second XPath expression is the  
1176 location path to the requested XML element and the first XPath expression is the literal value "md:record".  
1177 The "xpath-node-match" function evaluates to "True" if the requested XML element is below the  
1178 "md:record" element.

1179 [f68] - [f82] The `<AnyOf>` element contains a **disjunctive sequence** of `<AllOf>` elements. In this case,  
1180 there is just one `<AllOf>` element.

1181 [f69] - [f81] The `<AllOf>` element contains a **conjunctive sequence** of `<Match>` elements. In this case,  
1182 there is just one `<Match>` element.

1183 [f70] - [f80] The <Match> element compares its first and second child elements according to the matching  
1184 function. The match is positive if the value of the first argument matches any of the values selected by  
1185 the second argument. In this case, the value of the action-id **action attribute** in the request **context** is  
1186 compared with the literal value "read".

1187 [f84] - [f86] The <Condition> element. A **condition** must evaluate to "True" for the **rule** to be  
1188 applicable. This **condition** contains a reference to a variable definition defined elsewhere in the **policy**.

#### 1189 4.2.4.2 Rule 2

1190 **Rule 2** illustrates the use of a mathematical function, i.e. the <Apply> element with functionId  
1191 "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration" to calculate the date of the patient's  
1192 sixteenth birthday. It also illustrates the use of **predicate** expressions, with the functionId  
1193 "urn:oasis:names:tc:xacml:1.0:function:and". This example has one function embedded in the  
1194 <Condition> element and another one referenced in a <VariableDefinition> element.

```
1195 [g1] <?xml version="1.0" encoding="UTF-8"?>
1196 [g2] <Policy
1197 [g3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1198 [g4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1199 [g5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1200 [g6]   xmlns:xf="http://www.w3.org/2005/xfpath-functions"
1201 [g7]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1202 [g8]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"
1203 [g9]   Version="1.0"
1204 [g10]  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1205      algorithm:deny-overrides">
1206 [g11]  <PolicyDefaults>
1207 [g12]    <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1208 [g13]  </PolicyDefaults>
1209 [g14]  <Target/>
1210 [g15]  <VariableDefinition VariableId="17590035">
1211 [g16]    <Apply
1212 [g17]      FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal">
1213 [g18]      <Apply
1214 [g19]        FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
1215 [g20]          <AttributeDesignator
1216 [g21]            MustBePresent="false"
1217 [g22]            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
1218 [g23]            AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"
1219 [g24]            DataType="http://www.w3.org/2001/XMLSchema#date"/>
1220 [g25]          </Apply>
1221 [g26]        <Apply
1222 [g27]          FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration">
1223 [g28]            <Apply
1224 [g29]              FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
1225 [g30]                <AttributeSelector
1226 [g31]                  MustBePresent="false"
1227 [g32]                  Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1228 [g33]                  Path="md:record/md:patient/md:patientDoB/text()"
1229 [g34]                  DataType="http://www.w3.org/2001/XMLSchema#date"/>
1230 [g35]                </Apply>
1231 [g36]                <AttributeValue
1232 [g37]                  DataType="http://www.w3.org/2001/XMLSchema#yearMonthDuration"
1233 [g38]                  >P16Y</AttributeValue>
1234 [g39]              </Apply>
1235 [g40]            </Apply>
1236 [g41]          </VariableDefinition>
1237 [g42]  <Rule
1238 [g43]    RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"
1239 [g44]    Effect="Permit">
1240 [g45]    <Description>
1241 [g46]      A person may read any medical record in the
1242 [g47]      http://www.med.example.com/records.xsd namespace
1243 [g48]      for which he or she is the designated parent or guardian,
1244 [g49]      and for which the patient is under 16 years of age
1245 [g50]    </Description>
1246 [g51]    <Target>
1247 [g52]      <AnyOf>
1248 [g53]        <AllOf>
```

```

1249 [g54] <Match
1250 [g55]   MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1251 [g56]   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1252 [g57]     >urn:example:med:schemas:record</AttributeValue>
1253 [g58]   <AttributeDesignator
1254 [g59]     MustBePresent="false"
1255 [g60]     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1256 [g61]   AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1257 [g62]   DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1258 [g63] </Match>
1259 [g64] <Match
1260 [g65]   MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1261 [g66]   <AttributeValue
1262 [g67]     DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1263 [g68]   XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1264 [g69]   >md:record</AttributeValue>
1265 [g70]   <AttributeDesignator
1266 [g71]     MustBePresent="false"
1267 [g72]     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1268 [g73]     AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1269 [g74]     DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1270 [g75]   </Match>
1271 [g76] </AllOf>
1272 [g77] </AnyOf>
1273 [g78] <AnyOf>
1274 [g79]   <AllOf>
1275 [g80]     <Match
1276 [g81]       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1277 [g82]       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1278 [g83]         >read</AttributeValue>
1279 [g84]       <AttributeDesignator
1280 [g85]         MustBePresent="false"
1281 [g86]         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1282 [g87]         AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1283 [g88]         DataType="http://www.w3.org/2001/XMLSchema#string"/>
1284 [g89]       </Match>
1285 [g90]     </AllOf>
1286 [g91]   </AnyOf>
1287 [g92] </Target>
1288 [g93] <Condition>
1289 [g94]   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
1290 [g95]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1291 [g96]       <Apply
1292 [g97]         FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1293 [g98]           <AttributeDesignator
1294 [g99]             MustBePresent="false"
1295 [g100]           Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1296 [g101]           AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:parent-
guardian-id"
1297 [g102]           DataType="http://www.w3.org/2001/XMLSchema#string"/>
1298 [g103]         </Apply>
1299 [g104]       <Apply
1300 [g105]         FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1301 [g106]           <AttributeSelector
1302 [g107]             MustBePresent="false"
1303 [g108]             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1304 [g109]             Path="md:record/md:parentGuardian/md:parentGuardianId/text()"
1305 [g110]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1306 [g111]           </Apply>
1307 [g112]         </Apply>
1308 [g113]       </VariableReference VariableId="17590035"/>
1309 [g114]     </Apply>
1310 [g115]   </Condition>
1311 [g116] </Rule>
1312 [g117] </Policy>

```

1314 [g15] - [g41] The <VariableDefinition> element contains part of the **condition** (i.e. is the patient  
1315 under 16 years of age?). The patient is under 16 years of age if the current date is less than the date  
1316 computed by adding 16 to the patient's date of birth.

1317 [g16] - [g40] "urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal" is used to compare the two date  
1318 arguments.

1319 [g18] - [g25] The first date argument uses "urn:oasis:names:tc:xacml:1.0:function:date-one-and-only" to  
 1320 ensure that the **bag** of values selected by its argument contains exactly one value of type  
 1321 "http://www.w3.org/2001/XMLSchema#date".

1322 [g20] The current date is evaluated by selecting the "urn:oasis:names:tc:xacml:1.0:environment:current-  
 1323 date" **environment attribute**.

1324 [g26] - [g39] The second date argument uses "urn:oasis:names:tc:xacml:1.0:function:date-add-  
 1325 yearMonthDuration" to compute the date of the patient's sixteenth birthday by adding 16 years to the  
 1326 patient's date of birth. The first of its arguments is of type "http://www.w3.org/2001/XMLSchema#date"  
 1327 and the second is of type "http://www.w3.org/TR/2007/REC-xpath-functions-20070123/#dt-  
 1328 yearMonthDuration".

1329 [g30] The <AttributeSelector> element selects the patient's date of birth by taking the XPath  
 1330 expression over the **resource** content.

1331 [g36] - [g38] Year Month Duration of 16 years.

1332 [g51] - [g92] **Rule** declaration and **rule target**. See **Rule 1** in Section 4.2.4.1 for the detailed explanation  
 1333 of these elements.

1334 [g93] - [g115] The <Condition> element. The **condition** must evaluate to "True" for the **rule** to be  
 1335 applicable. This **condition** evaluates the truth of the statement: the requestor is the designated parent or  
 1336 guardian and the patient is under 16 years of age. It contains one embedded <Apply> element and one  
 1337 referenced <VariableDefinition> element.

1338 [g94] The **condition** uses the "urn:oasis:names:tc:xacml:1.0:function:and" function. This is a Boolean  
 1339 function that takes one or more Boolean arguments (2 in this case) and performs the logical "AND"  
 1340 operation to compute the truth value of the expression.

1341 [g95] - [g112] The first part of the **condition** is evaluated (i.e. is the requestor the designated parent or  
 1342 guardian?). The function is "urn:oasis:names:tc:xacml:1.0:function:string-equal" and it takes two  
 1343 arguments of type "http://www.w3.org/2001/XMLSchema#string".

1344 [g96] designates the first argument. Since "urn:oasis:names:tc:xacml:1.0:function:string-equal" takes  
 1345 arguments of type "http://www.w3.org/2001/XMLSchema#string",  
 1346 "urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" is used to ensure that the **subject attribute**  
 1347 "urn:oasis:names:tc:xacml:3.0:example:attribute:parent-guardian-id" in the request **context** contains  
 1348 exactly one value.

1349 [g98] designates the first argument. The value of the **subject attribute**  
 1350 "urn:oasis:names:tc:xacml:3.0:example:attribute:parent-guardian-id" is selected from the request **context**  
 1351 using the <AttributeDesignator> element.

1352 [g104] As above, the "urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" is used to ensure that  
 1353 the **bag** of values selected by its argument contains exactly one value of type  
 1354 "http://www.w3.org/2001/XMLSchema#string".

1355 [g106] The second argument selects the value of the <md:parentGuardianId> element from the  
 1356 **resource** content using the <AttributeSelector> element. This element contains a free-form XPath  
 1357 expression, pointing into the <Content> element of the resource category. Note that all namespace  
 1358 prefixes in the XPath expression are resolved with standard namespace declarations. The  
 1359 AttributeSelector evaluates to the **bag** of values of type  
 1360 "http://www.w3.org/2001/XMLSchema#string".

1361 [g113] references the <VariableDefinition> element, where the second part of the **condition** is  
 1362 defined.

#### 1363 4.2.4.3 Rule 3

1364 **Rule 3** illustrates the use of an **obligation** expression.

```

1365 [h1] <?xml version="1.0" encoding="UTF-8"?>
1366 [h2] <Policy
1367 [h3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1368 [h4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1369 [h5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

```

1370 [h6]      xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
1371 http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
1372 [h7]      xmlns:md="http://www.med.example.com/schemas/record.xsd"
1373 [h8]      PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:3"
1374 [h9]      Version="1.0"
1375 [h10]     RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1376 algorithm:deny-overrides">
1377 [h11]     <Description>
1378 [h12]       Policy for any medical record in the
1379 [h13]       http://www.med.example.com/schemas/record.xsd namespace
1380 [h14]     </Description>
1381 [h15]     <PolicyDefaults>
1382 [h16]       <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1383 [h17]     </PolicyDefaults>
1384 [h18]     <Target>
1385 [h19]       <AnyOf>
1386 [h20]         <AllOf>
1387 [h21]           <Match
1388 [h22]             MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1389 [h23]               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1390 [h24]                 >urn:example:med:schemas:record</AttributeValue>
1391 [h25]               <AttributeDesignator
1392 [h26]                 MustBePresent="false"
1393 [h27]                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1394 [h28]                 AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1395 [h29]                 DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1396 [h30]             </Match>
1397 [h31]           </AllOf>
1398 [h32]         </AnyOf>
1399 [h33]       </Target>
1400 [h34]     <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:3"
1401 [h35]       Effect="Permit">
1402 [h36]       <Description>
1403 [h37]         A physician may write any medical element in a record
1404 [h38]         for which he or she is the designated primary care
1405 [h39]         physician, provided an email is sent to the patient
1406 [h40]       </Description>
1407 [h41]       <Target>
1408 [h42]         <AnyOf>
1409 [h43]           <AllOf>
1410 [h44]             <Match
1411 [h45]               MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1412 [h46]                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1413 [h47]                   >physician</AttributeValue>
1414 [h48]                 <AttributeDesignator
1415 [h49]                   MustBePresent="false"
1416 [h50]                   Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1417 [h51]                   AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
1418 [h52]                   DataType="http://www.w3.org/2001/XMLSchema#string"/>
1419 [h53]               </Match>
1420 [h54]             </AllOf>
1421 [h55]           </AnyOf>
1422 [h56]         </AnyOf>
1423 [h57]       <AllOf>
1424 [h58]         <Match
1425 [h59]           MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1426 [h60]             <AttributeValue
1427 [h61]               DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1428 [h62]             XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1429 [h63]             >md:record/md:medical</AttributeValue>
1430 [h64]             <AttributeDesignator
1431 [h65]               MustBePresent="false"
1432 [h66]               Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1433 [h67]               AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1434 [h68]               DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1435 [h69]             </Match>
1436 [h70]           </AllOf>
1437 [h71]         </AnyOf>
1438 [h72]       </AnyOf>
1439 [h73]     <AllOf>
1440 [h74]       <Match
1441 [h75]         MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1442 [h76]         <AttributeValue

```

```

1443 [h77]         DataType="http://www.w3.org/2001/XMLSchema#string"
1444 [h78]         >write</AttributeValue>
1445 [h79]         <AttributeDesignator
1446 [h80]             MustBePresent="false"
1447 [h81]             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1448 [h82]             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1449 [h83]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1450 [h84]         </Match>
1451 [h85]         </AllOf>
1452 [h86]         </AnyOf>
1453 [h87]         </Target>
1454 [h88]         <Condition>
1455 [h89]             <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1456 [h90]                 <Apply
1457 [h91]                     FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1458 [h92]                         <AttributeDesignator
1459 [h93]                             MustBePresent="false"
1460 [h94]                             Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1461 [h95]                             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:physician-id"
1462 [h96]                             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1463 [h97]                         </Apply>
1464 [h98]                         <Apply
1465 [h99]                             FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1466 [h100]                                 <AttributeSelector
1467 [h101]                                     MustBePresent="false"
1468 [h102]                                     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1469 [h103]                                     Path="md:record/md:primaryCarePhysician/md:registrationID/text()"
1470 [h104]                                     DataType="http://www.w3.org/2001/XMLSchema#string"/>
1471 [h105]                                 </Apply>
1472 [h106]                             </Apply>
1473 [h107]                         </Condition>
1474 [h108]                     </Rule>
1475 [h109]                 <ObligationExpressions>
1476 [h110]                     <ObligationExpression
1477 [h111]                         ObligationId="urn:oasis:names:tc:xacml:example:obligation:email"
1478 [h112]                         FulfillOn="Permit">
1479 [h113]                             <AttributeAssignmentExpression
1480 [h114]                                 AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:mailto">
1481 [h115]                                     <AttributeSelector
1482 [h116]                                         MustBePresent="true"
1483 [h117]                                         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1484 [h118]                                         Path="md:record/md:patient/md:patientContact/md:email"
1485 [h119]                                         DataType="http://www.w3.org/2001/XMLSchema#string"/>
1486 [h120]                                     </AttributeAssignmentExpression>
1487 [h121]                                 <AttributeAssignmentExpression
1488 [h122]                                     AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
1489 [h123]                                         <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1490 [h124]                                             >Your medical record has been accessed by:</AttributeValue>
1491 [h125]                                         </AttributeAssignmentExpression>
1492 [h126]                                     <AttributeAssignmentExpression
1493 [h127]                                         AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
1494 [h128]                                             <AttributeDesignator
1495 [h129]                                                 MustBePresent="false"
1496 [h130]                                                 Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1497 [h131]                                                 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
1498 [h132]                                                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1499 [h133]                                             </AttributeAssignmentExpression>
1500 [h134]                                         </AttributeAssignmentExpression>
1501 [h135]                                     </ObligationExpression>
1502 [h136]                                 </ObligationExpressions>
1503 [h137]                             </ObligationExpression>
1504 [h138]                         </Policy>

```

1503 [h2] - [h10] The <Policy> element includes standard namespace declarations as well as **policy** specific  
1504 parameters, such as PolicyId and RuleCombiningAlgId.

1505 [h8] **Policy** identifier. This parameter allows the **policy** to be referenced by a **policy set**.

1506 [h10] The **Rule-combining algorithm** identifies the algorithm for combining the outcomes of **rule**  
1507 evaluation.

1508 [h11] - [h14] Free-form description of the **policy**.

1509 [h18] - [h33] **Policy target**. The **policy target** defines a set of applicable **decision requests**. The  
1510 structure of the <Target> element in the <Policy> is identical to the structure of the <Target>

1511 element in the <Rule>. In this case, the **policy target** is the set of all XML **resources** that conform to  
1512 the namespace "urn:example:med:schemas:record".

1513 [h34] - [h108] The only <Rule> element included in this <Policy>. Two parameters are specified in the  
1514 **rule** header: RuleId and Effect.

1515 [h41] - [h87] The **rule target** further constrains the **policy target**.

1516 [h44] - [h53] The <Match> element targets the **rule** at **subjects** whose  
1517 "urn:oasis:names:tc:xacml:3.0:example:attribute:role" **subject attribute** is equal to "physician".

1518 [h58] - [h69] The <Match> element targets the **rule** at **resources** that match the XPath expression  
1519 "md:record/md:medical".

1520 [h74] - [h84] The <Match> element targets the **rule** at **actions** whose  
1521 "urn:oasis:names:tc:xacml:1.0:action:action-id" **action attribute** is equal to "write".

1522 [h88] - [h107] The <Condition> element. For the **rule** to be applicable to the **decision request**, the  
1523 **condition** must evaluate to "True". This **condition** compares the value of the  
1524 "urn:oasis:names:tc:xacml:3.0:example:attribute:physician-id" **subject attribute** with the value of the  
1525 <registrationId> element in the medical record that is being accessed.

1526 [h109] - [h134] The <ObligationExpressions> element. **Obligations** are a set of operations that  
1527 must be performed by the **PEP** in conjunction with an **authorization decision**. An **obligation** may be  
1528 associated with a "Permit" or "Deny" **authorization decision**. The element contains a single **obligation**  
1529 expression, which will be evaluated into an obligation when the policy is evaluated.

1530 [h110] - [h133] The <ObligationExpression> element consists of the ObligationId attribute, the  
1531 **authorization decision** value for which it must be fulfilled, and a set of **attribute** assignments.

1532 [h110] The ObligationId attribute identifies the **obligation**. In this case, the **PEP** is required to send  
1533 email.

1534 [h111] The FulfillOn attribute defines the **authorization decision** value for which the **obligation**  
1535 derived from the **obligation** expression must be fulfilled. In this case, the **obligation** must be fulfilled  
1536 when **access** is permitted.

1537 [h112] - [h119] The first parameter indicates where the **PEP** will find the email address in the **resource**.  
1538 The **PDP** will evaluate the <AttributeSelector> and return the result to the **PEP** inside the resulting  
1539 **obligation**.

1540 [h120] - [h123] The second parameter contains literal text for the email body.

1541 [h125] - [h132] The third parameter indicates where the **PEP** will find further text for the email body in the  
1542 **resource**. The **PDP** will evaluate the <AttributeDesignator> and return the result to the **PEP** inside  
1543 the resulting **obligation**.

#### 1544 4.2.4.4 Rule 4

1545 **Rule 4** illustrates the use of the "Deny" **Effect** value, and a <Rule> with no <Condition> element.

```

1546 [i1] <?xml version="1.0" encoding="UTF-8"?>
1547 [i2] <Policy
1548 [i3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1549 [i4]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1550 [i5]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1551 [i6]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:4"
1552 [i7]   Version="1.0"
1553 [i8]   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1554   algorithm:deny-overrides">
1555 [i9]   <PolicyDefaults>
1556 [i10]     <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1557 [i11]   </PolicyDefaults>
1558 [i12]   <Target/>
1559 [i13]   <Rule
1560 [i14]     RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:4"
1561 [i15]     Effect="Deny">
1562 [i16]     <Description>
1563 [i17]       An Administrator shall not be permitted to read or write

```

```

1564 [i18] medical elements of a patient record in the
1565 [i19] http://www.med.example.com/records.xsd namespace.
1566 [i20] </Description>
1567 [i21] <Target>
1568 [i22] <AnyOf>
1569 [i23] <AllOf>
1570 [i24] <Match
1571 [i25] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1572 [i26] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1573 [i27] >administrator</AttributeValue>
1574 [i28] <AttributeDesignator
1575 [i29] MustBePresent="false"
1576 [i30] Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1577 [i31] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
1578 [i32] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1579 [i33] </Match>
1580 [i34] </AllOf>
1581 [i35] </AnyOf>
1582 [i36] <AnyOf>
1583 [i37] <AllOf>
1584 [i38] <Match
1585 [i39] MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1586 [i40] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1587 [i41] >urn:example:med:schemas:record</AttributeValue>
1588 [i42] <AttributeDesignator
1589 [i43] MustBePresent="false"
1590 [i44] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1591 [i45] AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1592 [i46] DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1593 [i47] </Match>
1594 [i48] <Match
1595 [i49] MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1596 [i50] <AttributeValue
1597 [i51] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1598 [i52] XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1599 [i53] >md:record/md:medical</AttributeValue>
1600 [i54] <AttributeDesignator
1601 [i55] MustBePresent="false"
1602 [i56] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1603 [i57] AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1604 [i58] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1605 [i59] </Match>
1606 [i60] </AllOf>
1607 [i61] </AnyOf>
1608 [i62] <AnyOf>
1609 [i63] <AllOf>
1610 [i64] <Match
1611 [i65] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1612 [i66] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1613 [i67] >read</AttributeValue>
1614 [i68] <AttributeDesignator
1615 [i69] MustBePresent="false"
1616 [i70] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1617 [i71] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1618 [i72] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1619 [i73] </Match>
1620 [i74] </AllOf>
1621 [i75] <AllOf>
1622 [i76] <Match
1623 [i77] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1624 [i78] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1625 [i79] >write</AttributeValue>
1626 [i80] <AttributeDesignator
1627 [i81] MustBePresent="false"
1628 [i82] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1629 [i83] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1630 [i84] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1631 [i85] </Match>
1632 [i86] </AllOf>
1633 [i87] </AnyOf>
1634 [i88] </Target>
1635 [i89] </Rule>
1636 [i90] </Policy>

```



1637 [i13] - [i15] The <Rule> element declaration.

1638 [i15] **Rule Effect**. Every **rule** that evaluates to “True” emits the **rule effect** as its value. This **rule**

1639 **Effect** is “Deny” meaning that according to this **rule**, **access** must be denied when it evaluates to

1640 “True”.

1641 [i16] - [i20] Free form description of the **rule**.

1642 [i21] - [i88] **Rule target**. The **Rule target** defines the set of **decision requests** that are applicable to the

1643 **rule**.

1644 [i24] - [i33] The <Match> element targets the **rule** at **subjects** whose

1645 “urn:oasis:names:tc:xacml:3.0:example:attribute:role” **subject attribute** is equal to “administrator”.

1646 [i36] - [i61] The <AnyOf> element contains one <AllOf> element, which (in turn) contains two <Match>

1647 elements. The **target** matches if the **resource** identified by the request **context** matches both **resource**

1648 match criteria.

1649 [i38] - [i47] The first <Match> element targets the **rule** at **resources** whose

1650 “urn:oasis:names:tc:xacml:2.0:resource:target-namespace” **resource attribute** is equal to

1651 “urn:example:med:schemas:record”.

1652 [i48] - [i59] The second <Match> element targets the **rule** at XML elements that match the XPath

1653 expression “/md:record/md:medical”.

1654 [i62] - [i87] The <AnyOf> element contains two <AllOf> elements, each of which contains one <Match>

1655 element. The **target** matches if the **action** identified in the request **context** matches either of the **action**

1656 match criteria.

1657 [i64] - [i85] The <Match> elements **target** the **rule** at **actions** whose

1658 “urn:oasis:names:tc:xacml:1.0:action:action-id” **action attribute** is equal to “read” or “write”.

1659 This **rule** does not have a <Condition> element.

#### 1660 4.2.4.5 Example PolicySet

1661 This section uses the examples of the previous sections to illustrate the process of combining **policies**.

1662 The **policy** governing read **access** to medical elements of a record is formed from each of the four **rules**

1663 described in Section 4.2.3. In plain language, the combined **rule** is:

- 1664 • Either the requestor is the patient; or
- 1665 • the requestor is the parent or guardian and the patient is under 16; or
- 1666 • the requestor is the primary care physician and a notification is sent to the patient; and
- 1667 • the requestor is not an administrator.

1668 The following **policy set** illustrates the combined **policies**. **Policy 3** is included by reference and **policy**

1669 **2** is explicitly included.

```

1670 [j1] <?xml version="1.0" encoding="UTF-8"?>
1671 [j2] <PolicySet
1672 [j3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1673 [j4]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1674 [j5]   PolicySetId="urn:oasis:names:tc:xacml:3.0:example:policysetid:1"
1675 [j6]   Version="1.0"
1676 [j7]   PolicyCombiningAlgId=
1677 [j8]   "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides">
1678 [j9]   <Description>
1679 [j10]     Example policy set.
1680 [j11]   </Description>
1681 [j12]   <Target>
1682 [j13]     <AnyOf>
1683 [j14]       <AllOf>
1684 [j15]         <Match
1685 [j16]           MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1686 [j17]             <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1687 [j18]               >urn:example:med:schema:records</AttributeValue>
1688 [j19]             <AttributeDesignator
1689 [j20]               MustBePresent="false"

```

```

1690 [j21]         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1691 [j22]         AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1692 [j23]         DataType="http://www.w3.org/2001/XMLSchema#string"/>
1693 [j24]         </Match>
1694 [j25]         </AllOf>
1695 [j26]         </AnyOf>
1696 [j27]     </Target>
1697 [j28]     <PolicyIdReference>
1698 [j29]         urn:oasis:names:tc:xacml:3.0:example:policyid:3
1699 [j30]     </PolicyIdReference>
1700 [j31]     <Policy>
1701 [j32]         PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"
1702 [j33]         RuleCombiningAlgId=
1703 [j34]             "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides"
1704 [j35]         Version="1.0">
1705 [j36]         <Target/>
1706 [j37]         <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"
1707 [j38]             Effect="Permit">
1708 [j39]         </Rule>
1709 [j40]         <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"
1710 [j41]             Effect="Permit">
1711 [j42]         </Rule>
1712 [j43]         <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:4"
1713 [j44]             Effect="Deny">
1714 [j45]         </Rule>
1715 [j46]     </Policy>
1716 [j47] </PolicySet>

```

1717 [j2] - [j8] The <PolicySet> element declaration. Standard XML namespace declarations are included.

1718 [j5] The PolicySetId attribute is used for identifying this **policy set** for possible inclusion in another **policy set**.

1720 [j7] - [j8] The **policy-combining algorithm** identifier. **Policies** and **policy sets** in this **policy set** are combined according to the specified **policy-combining algorithm** when the **authorization decision** is computed.

1723 [j9] - [j11] Free form description of the **policy set**.

1724 [j12] - [j27] The **policy set** <Target> element defines the set of **decision requests** that are applicable to this <PolicySet> element.

1726 [j28] - [j30] PolicyIdReference includes a **policy** by id.

1727 [j31] - [j46] **Policy 2** is explicitly included in this **policy set**. The **rules** in **Policy 2** are omitted for clarity.

## 1728 5 Syntax (normative, with the exception of the 1729 schema fragments)

### 1730 5.1 Element <PolicySet>

1731 The <PolicySet> element is a top-level element in the XACML *policy* schema. <PolicySet> is an  
1732 aggregation of other *policy sets* and *policies*. *Policy sets* MAY be included in an enclosing  
1733 <PolicySet> element either directly using the <PolicySet> element or indirectly using the  
1734 <PolicySetIdReference> element. *Policies* MAY be included in an enclosing <PolicySet>  
1735 element either directly using the <Policy> element or indirectly using the <PolicyIdReference>  
1736 element.

1737 A <PolicySet> element may be evaluated, in which case the evaluation procedure defined in Section  
1738 7.13 SHALL be used.

1739 If a <PolicySet> element contains references to other *policy sets* or *policies* in the form of URLs, then  
1740 these references MAY be resolvable.

1741 *Policy sets* and *policies* included in a <PolicySet> element MUST be combined using the algorithm  
1742 identified by the PolicyCombiningAlgId attribute. <PolicySet> is treated exactly like a <Policy>  
1743 in all *policy-combining algorithms*.

1744 A <PolicySet> element MAY contain a <PolicyIssuer> element. The interpretation of the  
1745 <PolicyIssuer> element is explained in the separate administrative *policy* profile [XACMLAdmin].

1746 The <Target> element defines the applicability of the <PolicySet> element to a set of *decision*  
1747 *requests*. If the <Target> element within the <PolicySet> element matches the request *context*,  
1748 then the <PolicySet> element MAY be used by the *PDP* in making its *authorization decision*. See  
1749 Section 7.13.

1750 The <ObligationExpressions> element contains a set of *obligation* expressions that MUST be  
1751 evaluated into *obligations* by the *PDP* and the resulting *obligations* MUST be fulfilled by the *PEP* in  
1752 conjunction with the *authorization decision*. If the *PEP* does not understand or cannot fulfill any of the  
1753 *obligations*, then it MUST act according to the PEP bias. See Section 7.2 and 7.18.

1754 The <AdviceExpressions> element contains a set of *advice* expressions that MUST be evaluated into  
1755 *advice* by the *PDP*. The resulting *advice* MAY be safely ignored by the *PEP* in conjunction with the  
1756 *authorization decision*. See Section 7.18.

1757

```
1758 <xs:element name="PolicySet" type="xacml:PolicySetType"/>
1759 <xs:complexType name="PolicySetType">
1760   <xs:sequence>
1761     <xs:element ref="xacml:Description" minOccurs="0"/>
1762     <xs:element ref="xacml:PolicyIssuer" minOccurs="0"/>
1763     <xs:element ref="xacml:PolicySetDefaults" minOccurs="0"/>
1764     <xs:element ref="xacml:Target"/>
1765     <xs:choice minOccurs="0" maxOccurs="unbounded">
1766       <xs:element ref="xacml:PolicySet"/>
1767       <xs:element ref="xacml:Policy"/>
1768       <xs:element ref="xacml:PolicySetIdReference"/>
1769       <xs:element ref="xacml:PolicyIdReference"/>
1770       <xs:element ref="xacml:CombinerParameters"/>
1771       <xs:element ref="xacml:PolicyCombinerParameters"/>
1772       <xs:element ref="xacml:PolicySetCombinerParameters"/>
1773     </xs:choice>
1774     <xs:element ref="xacml:ObligationExpressions" minOccurs="0"/>
1775     <xs:element ref="xacml:AdviceExpressions" minOccurs="0"/>
1776   </xs:sequence>
```

```

1777 <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
1778 <xs:attribute name="Version" type="xacml:VersionType" use="required"/>
1779 <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
1780 <xs:attribute name="MaxDelegationDepth" type="xs:integer" use="optional"/>
1781 </xs:complexType>

```

1782 The <PolicySet> element is of PolicySetType complex type.

1783 The <PolicySet> element contains the following attributes and elements:

1784 PolicySetId [Required]

1785 **Policy set** identifier. It is the responsibility of the **PAP** to ensure that no two **policies** visible to  
1786 the **PDP** have the same identifier. This MAY be achieved by following a predefined URN or URI  
1787 scheme. If the **policy set** identifier is in the form of a URL, then it MAY be resolvable.

1788 Version [Required]

1789 The version number of the PolicySet.

1790 PolicyCombiningAlgId [Required]

1791 The identifier of the **policy-combining algorithm** by which the <PolicySet>,  
1792 <CombinerParameters>, <PolicyCombinerParameters> and  
1793 <PolicySetCombinerParameters> components MUST be combined. Standard **policy-**  
1794 **combining algorithms** are listed in Appendix Appendix C. Standard **policy-combining**  
1795 **algorithm** identifiers are listed in Section B.9.

1796 MaxDelegationDepth [Optional]

1797 If present, limits the depth of delegation which is authorized by this **policy set**. See the delegation  
1798 profile [XACMLAdmin].

1799 <Description> [Optional]

1800 A free-form description of the **policy set**.

1801 <PolicyIssuer> [Optional]

1802 **Attributes** of the **issuer** of the **policy set**.

1803 <PolicySetDefaults> [Optional]

1804 A set of default values applicable to the **policy set**. The scope of the <PolicySetDefaults>  
1805 element SHALL be the enclosing **policy set**.

1806 <Target> [Required]

1807 The <Target> element defines the applicability of a **policy set** to a set of **decision requests**.

1808 The <Target> element MAY be declared by the creator of the <PolicySet> or it MAY be computed  
1809 from the <Target> elements of the referenced <Policy> elements, either as an intersection or  
1810 as a union.

1811 <PolicySet> [Any Number]

1812 A **policy set** that is included in this **policy set**.

1813 <Policy> [Any Number]

1814 A **policy** that is included in this **policy set**.

1815 <PolicySetIdReference> [Any Number]

1816 A reference to a **policy set** that MUST be included in this **policy set**. If  
1817 <PolicySetIdReference> is a URL, then it MAY be resolvable.

1818 <PolicyIdReference> [Any Number]

1819 A reference to a **policy** that MUST be included in this **policy set**. If the  
1820 <PolicyIdReference> is a URL, then it MAY be resolvable.

1821 <ObligationExpressions> [Optional]  
 1822       Contains the set of <ObligationExpression> elements. See Section 7.18 for a description of  
 1823       how the set of **obligations** to be returned by the **PDP** shall be determined.

1824 <AdviceExpressions> [Optional]  
 1825       Contains the set of <AdviceExpression> elements. See Section 7.18 for a description of how  
 1826       the set of **advice** to be returned by the **PDP** shall be determined.

1827 <CombinerParameters> [Optional]  
 1828       Contains a sequence of <CombinerParameter> elements. The parameters apply to the  
 1829       combining algorithm as such and it is up to the specific combining algorithm to interpret them and  
 1830       adjust its behavior accordingly.

1831 <PolicyCombinerParameters> [Optional]  
 1832       Contains a sequence of <CombinerParameter> elements that are associated with a particular  
 1833       <Policy> or <PolicyIdReference> element within the <PolicySet>. It is up to the specific  
 1834       combining algorithm to interpret them and adjust its behavior accordingly.

1835 <PolicySetCombinerParameters> [Optional]  
 1836       Contains a sequence of <CombinerParameter> elements that are associated with a particular  
 1837       <PolicySet> or <PolicySetIdReference> element within the <PolicySet>. It is up to the  
 1838       specific combining algorithm to interpret them and adjust its behavior accordingly.

## 1839 5.2 Element <Description>

1840 The <Description> element contains a free-form description of the <PolicySet>, <Policy>,  
 1841 <Rule> or <Apply> element. The <Description> element is of xs:string simple type.

1842 

```
<xs:element name="Description" type="xs:string"/>
```

## 1843 5.3 Element <PolicyIssuer>

1844 The <PolicyIssuer> element contains **attributes** describing the issuer of the **policy** or **policy set**.  
 1845 The use of the **policy** issuer element is defined in a separate administration profile [**XACMLAdmin**]. A  
 1846 PDP which does not implement the administration profile **MUST** report an error or return an Indeterminate  
 1847 result if it encounters this element.

1848 

```
<xs:element name="PolicyIssuer" type="xacml:PolicyIssuerType"/>  

  1849 <xs:complexType name="PolicyIssuerType">  

  1850 <xs:sequence>  

  1851 <xs:element ref="xacml:Content" minOccurs="0"/>  

  1852 <xs:element ref="xacml:Attribute" minOccurs="0" maxOccurs="unbounded"/>  

  1853 </xs:sequence>  

  1854 </xs:complexType>
```

1855 The <PolicyIssuer> element is of PolicyIssuerType complex type.

1856 The <PolicyIssuer> element contains the following elements:

1857 <Content> [Optional]

1858       Free form XML describing the issuer. See Section 5.45.

1859 <Attribute> [Zero to many]

1860       An **attribute** of the issuer. See Section 5.46.

## 1861 5.4 Element <PolicySetDefaults>

1862 The <PolicySetDefaults> element SHALL specify default values that apply to the <PolicySet>  
 1863 element.

1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871

```
<xs:element name="PolicySetDefaults" type="xacml:DefaultsType"/>
<xs:complexType name="DefaultsType">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="xacml:XPathVersion">
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

1872 <PolicySetDefaults> element is of DefaultsType complex type.

1873 The <PolicySetDefaults> element contains the following elements:

1874 <XPathVersion> [Optional]

1875 Default XPath version.

## 1876 5.5 Element <XPathVersion>

1877 The <XPathVersion> element SHALL specify the version of the XPath specification to be used by  
1878 <AttributeSelector> elements and XPath-based functions in the **policy set** or **policy**.

1879

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

1880 The URI for the XPath 1.0 specification is "http://www.w3.org/TR/1999/REC-xpath-19991116".

1881 The URI for the XPath 2.0 specification is "http://www.w3.org/TR/2007/REC-xpath20-20070123".

1882 The <XPathVersion> element is REQUIRED if the XACML enclosing **policy set** or **policy** contains

1883 <AttributeSelector> elements or XPath-based functions.

## 1884 5.6 Element <Target>

1885 The <Target> element identifies the set of **decision requests** that the parent element is intended to  
1886 evaluate. The <Target> element SHALL appear as a child of a <PolicySet> and <Policy> element  
1887 and MAY appear as a child of a <Rule> element.

1888 The <Target> element SHALL contain a **conjunctive sequence** of <AnyOf> elements. For the parent

1889 of the <Target> element to be applicable to the **decision request**, there MUST be at least one positive

1890 match between each <AnyOf> element of the <Target> element and the corresponding section of the

1891 <Request> element.

1892  
1893  
1894  
1895  
1896  
1897

```
<xs:element name="Target" type="xacml:TargetType"/>
<xs:complexType name="TargetType">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="xacml:AnyOf"/>
  </xs:sequence>
</xs:complexType>
```

1898 The <Target> element is of TargetType complex type.

1899 The <Target> element contains the following elements:

1900 <AnyOf> [Zero to Many]

1901 Matching specification for **attributes** in the **context**. If this element is missing, then the **target**  
1902 SHALL match all **contexts**.

## 1903 5.7 Element <AnyOf>

1904 The <AnyOf> element SHALL contain a **disjunctive sequence** of <AllOf> elements.

1905  
1906  
1907  
1908

```
<xs:element name="AnyOf" type="xacml:AnyOfType"/>
<xs:complexType name="AnyOfType">
  <xs:sequence minOccurs="1" maxOccurs="unbounded">
    <xs:element ref="xacml:AllOf"/>
  </xs:sequence>
</xs:complexType>
```

1909 `</xs:sequence>`  
1910 `</xs:complexType>`

1911 The `<AnyOf>` element is of `AnyOfType` complex type.

1912 The `<AnyOf>` element contains the following elements:

1913 `<AllOf>` [One to Many, Required]

1914 See Section 5.8.

## 1915 5.8 Element `<AllOf>`

1916 The `<AllOf>` element SHALL contain a **conjunctive sequence** of `<Match>` elements.

```
1917 <xs:element name="AllOf" type="xacml:AllOfType"/>
1918 <xs:complexType name="AllOfType">
1919   <xs:sequence minOccurs="1" maxOccurs="unbounded">
1920     <xs:element ref="xacml:Match"/>
1921   </xs:sequence>
1922 </xs:complexType>
```

1923 The `<AllOf>` element is of `AllOfType` complex type.

1924 The `<AllOf>` element contains the following elements:

1925 `<Match>` [One to Many]

1926 A **conjunctive sequence** of individual matches of the **attributes** in the request **context** and the  
1927 embedded **attribute** values. See Section 5.9.

## 1928 5.9 Element `<Match>`

1929 The `<Match>` element SHALL identify a set of entities by matching **attribute** values in an

1930 `<Attributes>` element of the request **context** with the embedded **attribute** value.

```
1931 <xs:element name="Match" type="xacml:MatchType"/>
1932 <xs:complexType name="MatchType">
1933   <xs:sequence>
1934     <xs:element ref="xacml:AttributeValue"/>
1935     <xs:choice>
1936       <xs:element ref="xacml:AttributeDesignator"/>
1937       <xs:element ref="xacml:AttributeSelector"/>
1938     </xs:choice>
1939   </xs:sequence>
1940   <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
1941 </xs:complexType>
```

1942 The `<Match>` element is of `MatchType` complex type.

1943 The `<Match>` element contains the following attributes and elements:

1944 `MatchId` [Required]

1945 Specifies a matching function. The value of this attribute MUST be of type `xs:anyURI` with legal  
1946 values documented in Section 7.6.

1947 `<AttributeValue>` [Required]

1948 Embedded **attribute** value.

1949 `<AttributeDesignator>` [Required choice]

1950 MAY be used to identify one or more **attribute** values in an `<Attributes>` element of the  
1951 request **context**.

1952 `<AttributeSelector>` [Required choice]

1953 MAY be used to identify one or more **attribute** values in a <Content> element of the request  
1954 **context**.

## 1955 5.10 Element <PolicySetIdReference>

1956 The <PolicySetIdReference> element SHALL be used to reference a <PolicySet> element by id.  
1957 If <PolicySetIdReference> is a URL, then it MAY be resolvable to the <PolicySet> element.  
1958 However, the mechanism for resolving a **policy set** reference to the corresponding **policy set** is outside  
1959 the scope of this specification.

```
1960 <xs:element name="PolicySetIdReference" type="xacml:IdReferenceType"/>  
1961 <xs:complexType name="IdReferenceType">  
1962   <xs:simpleContent>  
1963     <xs:extension base="xs:anyURI">  
1964       <xs:attribute name="xacml:Version"  
1965         type="xacml:VersionMatchType" use="optional"/>  
1966       <xs:attribute name="xacml:EarliestVersion"  
1967         type="xacml:VersionMatchType" use="optional"/>  
1968       <xs:attribute name="xacml:LatestVersion"  
1969         type="xacml:VersionMatchType" use="optional"/>  
1970     </xs:extension>  
1971   </xs:simpleContent>  
1972 </xs:complexType>
```

1973 Element <PolicySetIdReference> is of xacml:IdReferenceType complex type.

1974 IdReferenceType extends the xs:anyURI type with the following attributes:

1975 Version [Optional]

1976 Specifies a matching expression for the version of the **policy set** referenced.

1977 EarliestVersion [Optional]

1978 Specifies a matching expression for the earliest acceptable version of the **policy set** referenced.

1979 LatestVersion [Optional]

1980 Specifies a matching expression for the latest acceptable version of the **policy set** referenced.

1981 The matching operation is defined in Section 5.13. Any combination of these attributes MAY be present  
1982 in a <PolicySetIdReference>. The referenced **policy set** MUST match all expressions. If none of  
1983 these attributes is present, then any version of the **policy set** is acceptable. In the case that more than  
1984 one matching version can be obtained, then the most recent one SHOULD be used.

## 1985 5.11 Element <PolicyIdReference>

1986 The <PolicyIdReference> element SHALL be used to reference a <Policy> element by id. If  
1987 <PolicyIdReference> is a URL, then it MAY be resolvable to the <Policy> element. However, the  
1988 mechanism for resolving a **policy** reference to the corresponding **policy** is outside the scope of this  
1989 specification.

```
1990 <xs:element name="PolicyIdReference" type="xacml:IdReferenceType"/>
```

1991 Element <PolicyIdReference> is of xacml:IdReferenceType complex type (see Section 5.10).

## 1992 5.12 Simple type VersionType

1993 Elements of this type SHALL contain the version number of the **policy** or **policy set**.

```
1994 <xs:simpleType name="VersionType">  
1995   <xs:restriction base="xs:string">  
1996     <xs:pattern value="(\d+\.)*\d+"/>  
1997   </xs:restriction>  
1998 </xs:simpleType>
```



1999 The version number is expressed as a sequence of decimal numbers, each separated by a period (.).  
2000 'd+' represents a sequence of one or more decimal digits.

### 2001 5.13 Simple type VersionMatchType

2002 Elements of this type SHALL contain a restricted regular expression matching a version number (see  
2003 Section 5.12). The expression SHALL match versions of a referenced *policy* or *policy set* that are  
2004 acceptable for inclusion in the referencing *policy* or *policy set*.

```
2005 <xs:simpleType name="VersionMatchType">  
2006   <xs:restriction base="xs:string">  
2007     <xs:pattern value="((\d+|\*)\.)*(\d+|\*|\+)" />  
2008   </xs:restriction>  
2009 </xs:simpleType>
```

2010 A version match is '.'-separated, like a version string. A number represents a direct numeric match. A '\*'  
2011 means that any single number is valid. A '+' means that any number, and any subsequent numbers, are  
2012 valid. In this manner, the following four patterns would all match the version string '1.2.3': '1.2.3', '1.\*.3',  
2013 '1.2.\*' and '1.+'

### 2014 5.14 Element <Policy>

2015 The <Policy> element is the smallest entity that SHALL be presented to the *PDP* for evaluation.

2016 A <Policy> element may be evaluated, in which case the evaluation procedure defined in Section 7.12  
2017 SHALL be used.

2018 The main components of this element are the <Target>, <Rule>, <CombinerParameters>,  
2019 <RuleCombinerParameters>, <ObligationExpressions> and <AdviceExpressions>  
2020 elements and the RuleCombiningAlgId attribute.

2021 A <Policy> element MAY contain a <PolicyIssuer> element. The interpretation of the  
2022 <PolicyIssuer> element is explained in the separate administrative *policy* profile [XACMLAdmin].

2023 The <Target> element defines the applicability of the <Policy> element to a set of *decision requests*.

2024 If the <Target> element within the <Policy> element matches the request *context*, then the  
2025 <Policy> element MAY be used by the *PDP* in making its *authorization decision*. See Section 7.12.

2026 The <Policy> element includes a sequence of choices between <VariableDefinition> and  
2027 <Rule> elements.

2028 *Rules* included in the <Policy> element MUST be combined by the algorithm specified by the  
2029 RuleCombiningAlgId attribute.

2030 The <ObligationExpressions> element contains a set of *obligation* expressions that MUST be  
2031 evaluated into *obligations* by the *PDP* and the resulting *obligations* MUST be fulfilled by the *PEP* in  
2032 conjunction with the *authorization decision*. If the *PEP* does not understand, or cannot fulfill, any of the  
2033 *obligations*, then it MUST act according to the PEP bias. See Section 7.2 and 7.18.

2034 The <AdviceExpressions> element contains a set of *advice* expressions that MUST be evaluated into  
2035 *advice* by the *PDP*. The resulting *advice* MAY be safely ignored by the *PEP* in conjunction with the  
2036 *authorization decision*. See Section 7.18.

```
2037 <xs:element name="Policy" type="xacml:PolicyType"/>  
2038 <xs:complexType name="PolicyType">  
2039   <xs:sequence>  
2040     <xs:element ref="xacml:Description" minOccurs="0"/>  
2041     <xs:element ref="xacml:PolicyIssuer" minOccurs="0"/>  
2042     <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>  
2043     <xs:element ref="xacml:Target"/>  
2044     <xs:choice maxOccurs="unbounded">  
2045       <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>  
2046       <xs:element ref="xacml:RuleCombinerParameters" minOccurs="0"/>  
2047       <xs:element ref="xacml:VariableDefinition"/>
```

```

2048         <xs:element ref="xacml:Rule"/>
2049     </xs:choice>
2050     <xs:element ref="xacml:ObligationExpressions" minOccurs="0"/>
2051     <xs:element ref="xacml:AdviceExpressions" minOccurs="0"/>
2052 </xs:sequence>
2053 <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
2054 <xs:attribute name="Version" type="xacml:VersionType" use="required"/>
2055 <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
2056 <xs:attribute name="MaxDelegationDepth" type="xs:integer" use="optional"/>
2057 </xs:complexType>

```

2058 The <Policy> element is of PolicyType complex type.

2059 The <Policy> element contains the following attributes and elements:

2060 PolicyId [Required]

2061 **Policy** identifier. It is the responsibility of the **PAP** to ensure that no two **policies** visible to the  
2062 **PDP** have the same identifier. This MAY be achieved by following a predefined URN or URI  
2063 scheme. If the **policy** identifier is in the form of a URL, then it MAY be resolvable.

2064 Version [Required]

2065 The version number of the **Policy**.

2066 RuleCombiningAlgId [Required]

2067 The identifier of the **rule-combining algorithm** by which the <Policy>,  
2068 <CombinerParameters> and <RuleCombinerParameters> components MUST be  
2069 combined. Standard **rule-combining algorithms** are listed in Appendix Appendix C. Standard  
2070 **rule-combining algorithm** identifiers are listed in Section B.9.

2071 MaxDelegationDepth [Optional]

2072 If present, limits the depth of delegation which is authorized by this **policy**. See the delegation  
2073 profile [XACMLAdmin].

2074 <Description> [Optional]

2075 A free-form description of the **policy**. See Section 5.2.

2076 <PolicyIssuer> [Optional]

2077 **Attributes** of the **issuer** of the **policy**.

2078 <PolicyDefaults> [Optional]

2079 Defines a set of default values applicable to the **policy**. The scope of the <PolicyDefaults>  
2080 element SHALL be the enclosing **policy**.

2081 <CombinerParameters> [Optional]

2082 A sequence of parameters to be used by the **rule-combining algorithm**. The parameters apply  
2083 to the combining algorithm as such and it is up to the specific combining algorithm to interpret  
2084 them and adjust its behavior accordingly.

2085 <RuleCombinerParameters> [Optional]

2086 A sequence of <RuleCombinerParameter> elements that are associated with a particular  
2087 <Rule> element within the <Policy>.. It is up to the specific combining algorithm to interpret  
2088 them and adjust its behavior accordingly.

2089 <Target> [Required]

2090 The <Target> element defines the applicability of a <Policy> to a set of **decision requests**.

2091 The <Target> element MAY be declared by the creator of the <Policy> element, or it MAY be  
2092 computed from the <Target> elements of the referenced <Rule> elements either as an  
2093 intersection or as a union.

- 2094 <VariableDefinition> [Any Number]  
 2095 Common function definitions that can be referenced from anywhere in a **rule** where an  
 2096 expression can be found.
- 2097 <Rule> [Any Number]  
 2098 A sequence of **rules** that MUST be combined according to the `RuleCombiningAlgId` attribute.  
 2099 **Rules** whose <Target> elements and conditions match the **decision request** MUST be  
 2100 considered. **Rules** whose <Target> elements or conditions do not match the **decision request**  
 2101 SHALL be ignored.
- 2102 <ObligationExpressions> [Optional]  
 2103 A **conjunctive sequence** of **obligation** expressions which MUST be evaluated into **obligations**  
 2104 by the PDP. The corresponding **obligations** MUST be fulfilled by the **PEP** in conjunction with  
 2105 the **authorization decision**. See Section 7.18 for a description of how the set of **obligations** to  
 2106 be returned by the **PDP** SHALL be determined. See section 7.2 about enforcement of  
 2107 **obligations**.
- 2108 <AdviceExpressions> [Optional]  
 2109 A **conjunctive sequence** of **advice** expressions which MUST evaluated into **advice** by the **PDP**.  
 2110 The corresponding **advice** provide supplementary information to the **PEP** in conjunction with the  
 2111 **authorization decision**. See Section 7.18 for a description of how the set of **advice** to be  
 2112 returned by the **PDP** SHALL be determined.

## 2113 5.15 Element <PolicyDefaults>

2114 The <PolicyDefaults> element SHALL specify default values that apply to the <Policy> element.

```
2115 <xs:element name="PolicyDefaults" type="xacml:DefaultsType"/>
2116 <xs:complexType name="DefaultsType">
2117   <xs:sequence>
2118     <xs:choice>
2119       <xs:element ref="xacml:XPathVersion" />
2120     </xs:choice>
2121   </xs:sequence>
2122 </xs:complexType>
```

- 2123 <PolicyDefaults> element is of DefaultsType complex type.  
 2124 The <PolicyDefaults> element contains the following elements:  
 2125 <XPathVersion> [Optional]  
 2126 Default XPath version.

## 2127 5.16 Element <CombinerParameters>

- 2128 The <CombinerParameters> element conveys parameters for a **policy**- or **rule-combining algorithm**.  
 2129 If multiple <CombinerParameters> elements occur within the same **policy** or **policy set**, they SHALL  
 2130 be considered equal to one <CombinerParameters> element containing the concatenation of all the  
 2131 sequences of <CombinerParameters> contained in all the aforementioned <CombinerParameters>  
 2132 elements, such that the order of occurrence of the <CombinerParameters> elements is preserved in the  
 2133 concatenation of the <CombinerParameter> elements.  
 2134 Note that none of the combining algorithms specified in XACML 3.0 is parameterized.

```
2135 <xs:element name="CombinerParameters" type="xacml:CombinerParametersType"/>
2136 <xs:complexType name="CombinerParametersType">
2137   <xs:sequence>
2138     <xs:element ref="xacml:CombinerParameter" minOccurs="0"
2139       maxOccurs="unbounded"/>
2140   </xs:sequence>
```

2141 `</xs:complexType>`

2142 The `<CombinerParameters>` element is of `CombinerParametersType` complex type.

2143 The `<CombinerParameters>` element contains the following elements:

2144 `<CombinerParameter>` [Any Number]

2145 A single parameter. See Section 5.17.

2146 Support for the `<CombinerParameters>` element is optional.

## 2147 5.17 Element `<CombinerParameter>`

2148 The `<CombinerParameter>` element conveys a single parameter for a *policy*- or *rule-combining algorithm*.

```
2150 <xs:element name="CombinerParameter" type="xacml:CombinerParameterType"/>
2151 <xs:complexType name="CombinerParameterType">
2152   <xs:sequence>
2153     <xs:element ref="xacml:AttributeValue"/>
2154   </xs:sequence>
2155   <xs:attribute name="ParameterName" type="xs:string" use="required"/>
2156 </xs:complexType>
```

2157 The `<CombinerParameter>` element is of `CombinerParameterType` complex type.

2158 The `<CombinerParameter>` element contains the following attributes:

2159 `ParameterName` [Required]

2160 The identifier of the parameter.

2161 `<AttributeValue>` [Required]

2162 The value of the parameter.

2163 Support for the `<CombinerParameter>` element is optional.

## 2164 5.18 Element `<RuleCombinerParameters>`

2165 The `<RuleCombinerParameters>` element conveys parameters associated with a particular *rule* within a *policy* for a *rule-combining algorithm*.

2167 Each `<RuleCombinerParameters>` element MUST be associated with a *rule* contained within the same *policy*. If multiple `<RuleCombinerParameters>` elements reference the same *rule*, they SHALL be considered equal to one `<RuleCombinerParameters>` element containing the concatenation of all the sequences of `<CombinerParameters>` contained in all the aforementioned `<RuleCombinerParameters>` elements, such that the order of occurrence of the `<RuleCombinerParameters>` elements is preserved in the concatenation of the `<CombinerParameter>` elements.

2174 Note that none of the *rule-combining algorithms* specified in XACML 3.0 is parameterized.

```
2175 <xs:element name="RuleCombinerParameters"
2176   type="xacml:RuleCombinerParametersType"/>
2177 <xs:complexType name="RuleCombinerParametersType">
2178   <xs:complexContent>
2179     <xs:extension base="xacml:CombinerParametersType">
2180       <xs:attribute name="RuleIdRef" type="xs:string"
2181         use="required"/>
2182     </xs:extension>
2183   </xs:complexContent>
2184 </xs:complexType>
```

2185 The `<RuleCombinerParameters>` element contains the following attribute:

2186 RuleIdRef [Required]

2187 The identifier of the <Rule> contained in the **policy**.

2188 Support for the <RuleCombinerParameters> element is optional, only if support for combiner  
2189 parameters is not implemented.

## 2190 5.19 Element <PolicyCombinerParameters>

2191 The <PolicyCombinerParameters> element conveys parameters associated with a particular **policy**  
2192 within a **policy set** for a **policy-combining algorithm**.

2193 Each <PolicyCombinerParameters> element MUST be associated with a **policy** contained within the  
2194 same **policy set**. If multiple <PolicyCombinerParameters> elements reference the same **policy**,  
2195 they SHALL be considered equal to one <PolicyCombinerParameters> element containing the  
2196 concatenation of all the sequences of <CombinerParameters> contained in all the aforementioned  
2197 <PolicyCombinerParameters> elements, such that the order of occurrence of the  
2198 <PolicyCombinerParameters> elements is preserved in the concatenation of the  
2199 <CombinerParameter> elements.

2200 Note that none of the **policy-combining algorithms** specified in XACML 3.0 is parameterized.

```
2201 <xs:element name="PolicyCombinerParameters"  
2202 type="xacml:PolicyCombinerParametersType"/>  
2203 <xs:complexType name="PolicyCombinerParametersType">  
2204 <xs:complexContent>  
2205 <xs:extension base="xacml:CombinerParametersType">  
2206 <xs:attribute name="PolicyIdRef" type="xs:anyURI"  
2207 use="required"/>  
2208 </xs:extension>  
2209 </xs:complexContent>  
2210 </xs:complexType>
```

2211 The <PolicyCombinerParameters> element is of PolicyCombinerParametersType complex  
2212 type.

2213 The <PolicyCombinerParameters> element contains the following attribute:

2214 PolicyIdRef [Required]

2215 The identifier of a <Policy> or the value of a <PolicyIdReference> contained in the **policy**  
2216 **set**.

2217 Support for the <PolicyCombinerParameters> element is optional, only if support for combiner  
2218 parameters is not implemented.

## 2219 5.20 Element <PolicySetCombinerParameters>

2220 The <PolicySetCombinerParameters> element conveys parameters associated with a particular  
2221 **policy set** within a **policy set** for a **policy-combining algorithm**.

2222 Each <PolicySetCombinerParameters> element MUST be associated with a **policy set** contained  
2223 within the same **policy set**. If multiple <PolicySetCombinerParameters> elements reference the  
2224 same **policy set**, they SHALL be considered equal to one <PolicySetCombinerParameters>  
2225 element containing the concatenation of all the sequences of <CombinerParameters> contained in all  
2226 the aforementioned <PolicySetCombinerParameters> elements, such that the order of occurrence  
2227 of the <PolicySetCombinerParameters> elements is preserved in the concatenation of the  
2228 <CombinerParameter> elements.

2229 Note that none of the **policy-combining algorithms** specified in XACML 3.0 is parameterized.

```
2230 <xs:element name="PolicySetCombinerParameters"  
2231 type="xacml:PolicySetCombinerParametersType"/>  
2232 <xs:complexType name="PolicySetCombinerParametersType">
```

```

2233 <xs:complexContent>
2234   <xs:extension base="xacml:CombinerParametersType">
2235     <xs:attribute name="PolicySetIdRef" type="xs:anyURI"
2236     use="required"/>
2237   </xs:extension>
2238 </xs:complexContent>
2239 </xs:complexType>

```

2240 The <PolicySetCombinerParameters> element is of PolicySetCombinerParametersType  
2241 complex type.

2242 The <PolicySetCombinerParameters> element contains the following attribute:

2243 PolicySetIdRef [Required]

2244       The identifier of a <PolicySet> or the value of a <PolicySetIdReference> contained in the  
2245       *policy set*.

2246 Support for the <PolicySetCombinerParameters> element is optional, only if support for combiner  
2247 parameters is not implemented.

## 2248 5.21 Element <Rule>

2249 The <Rule> element SHALL define the individual *rules* in the *policy*. The main components of this  
2250 element are the <Target>, <Condition>, <ObligationExpressions> and  
2251 <AdviceExpressions> elements and the Effect attribute.

2252 A <Rule> element may be evaluated, in which case the evaluation procedure defined in Section 7.10  
2253 SHALL be used.

```

2254 <xs:element name="Rule" type="xacml:RuleType"/>
2255 <xs:complexType name="RuleType">
2256   <xs:sequence>
2257     <xs:element ref="xacml:Description" minOccurs="0"/>
2258     <xs:element ref="xacml:Target" minOccurs="0"/>
2259     <xs:element ref="xacml:Condition" minOccurs="0"/>
2260     <xs:element ref="xacml:ObligationExpressions" minOccurs="0"/>
2261     <xs:element ref="xacml:AdviceExpressions" minOccurs="0"/>
2262   </xs:sequence>
2263   <xs:attribute name="RuleId" type="xs:string" use="required"/>
2264   <xs:attribute name="Effect" type="xacml:EffectType" use="required"/>
2265 </xs:complexType>

```

2266 The <Rule> element is of RuleType complex type.

2267 The <Rule> element contains the following attributes and elements:

2268 RuleId [Required]

2269       A string identifying this *rule*.

2270 Effect [Required]

2271       **Rule effect.** The value of this attribute is either "Permit" or "Deny".

2272 <Description> [Optional]

2273       A free-form description of the *rule*.

2274 <Target> [Optional]

2275       Identifies the set of *decision requests* that the <Rule> element is intended to evaluate. If this  
2276       element is omitted, then the **target** for the <Rule> SHALL be defined by the <Target> element  
2277       of the enclosing <Policy> element. See Section 7.7 for details.

2278 <Condition> [Optional]

2279       A *predicate* that MUST be satisfied for the *rule* to be assigned its Effect value.

2280 <ObligationExpressions> [Optional]

2281 A **conjunctive sequence** of **obligation** expressions which MUST be evaluated into **obligations**  
2282 by the PDP. The corresponding **obligations** MUST be fulfilled by the **PEP** in conjunction with  
2283 the **authorization decision**. See Section 7.18 for a description of how the set of **obligations** to  
2284 be returned by the **PDP** SHALL be determined. See section 7.2 about enforcement of  
2285 **obligations**.

2286 <AdviceExpressions> [Optional]

2287 A **conjunctive sequence** of **advice** expressions which MUST be evaluated into **advice** by the **PDP**.  
2288 The corresponding **advice** provide supplementary information to the **PEP** in conjunction with the  
2289 **authorization decision**. See Section 7.18 for a description of how the set of **advice** to be  
2290 returned by the **PDP** SHALL be determined.

## 2291 5.22 Simple type EffectType

2292 The EffectType simple type defines the values allowed for the Effect attribute of the <Rule> element  
2293 and for the FulfillOn attribute of the <ObligationExpression> and <AdviceExpression>  
2294 elements.

```
2295 <xs:simpleType name="EffectType">  
2296   <xs:restriction base="xs:string">  
2297     <xs:enumeration value="Permit"/>  
2298     <xs:enumeration value="Deny"/>  
2299   </xs:restriction>  
2300 </xs:simpleType>
```

## 2301 5.23 Element <VariableDefinition>

2302 The <VariableDefinition> element SHALL be used to define a value that can be referenced by a  
2303 <VariableReference> element. The name supplied for its VariableId attribute SHALL NOT occur  
2304 in the VariableId attribute of any other <VariableDefinition> element within the encompassing  
2305 **policy**. The <VariableDefinition> element MAY contain undefined <VariableReference>  
2306 elements, but if it does, a corresponding <VariableDefinition> element MUST be defined later in  
2307 the encompassing **policy**. <VariableDefinition> elements MAY be grouped together or MAY be  
2308 placed close to the reference in the encompassing **policy**. There MAY be zero or more references to  
2309 each <VariableDefinition> element.

```
2310 <xs:element name="VariableDefinition" type="xacml:VariableDefinitionType"/>  
2311 <xs:complexType name="VariableDefinitionType">  
2312   <xs:sequence>  
2313     <xs:element ref="xacml:Expression"/>  
2314   </xs:sequence>  
2315   <xs:attribute name="VariableId" type="xs:string" use="required"/>  
2316 </xs:complexType>
```

2317 The <VariableDefinition> element is of VariableDefinitionType complex type. The  
2318 <VariableDefinition> element has the following elements and attributes:

2319 <Expression> [Required]

2320 Any element of ExpressionType complex type.

2321 VariableId [Required]

2322 The name of the variable definition.

## 2323 5.24 Element <VariableReference>

2324 The <VariableReference> element is used to reference a value defined within the same  
2325 encompassing <Policy> element. The <VariableReference> element SHALL refer to the

2326 <VariableDefinition> element by **identifier equality** on the value of their respective VariableId  
2327 attributes. One and only one <VariableDefinition> MUST exist within the same encompassing  
2328 <Policy> element to which the <VariableReference> refers. There MAY be zero or more  
2329 <VariableReference> elements that refer to the same <VariableDefinition> element.

```
2330 <xs:element name="VariableReference" type="xacml:VariableReferenceType"  
2331 substitutionGroup="xacml:Expression"/>  
2332 <xs:complexType name="VariableReferenceType">  
2333 <xs:complexContent>  
2334 <xs:extension base="xacml:ExpressionType">  
2335 <xs:attribute name="VariableId" type="xs:string"  
2336 use="required"/>  
2337 </xs:extension>  
2338 </xs:complexContent>  
2339 </xs:complexType>
```

2340 The <VariableReference> element is of the VariableReferenceType complex type, which is of  
2341 the ExpressionType complex type and is a member of the <Expression> element substitution group.  
2342 The <VariableReference> element MAY appear any place where an <Expression> element occurs  
2343 in the schema.

2344 The <VariableReference> element has the following attribute:

2345 VariableId [Required]

2346 The name used to refer to the value defined in a <VariableDefinition> element.

## 2347 5.25 Element <Expression>

2348 The <Expression> element is not used directly in a **policy**. The <Expression> element signifies that  
2349 an element that extends the ExpressionType and is a member of the <Expression> element  
2350 substitution group SHALL appear in its place.

```
2351 <xs:element name="Expression" type="xacml:ExpressionType" abstract="true"/>  
2352 <xs:complexType name="ExpressionType" abstract="true"/>
```

2353 The following elements are in the <Expression> element substitution group:

2354 <Apply>, <AttributeSelector>, <AttributeValue>, <Function>, <VariableReference> and  
2355 <AttributeDesignator>.

## 2356 5.26 Element <Condition>

2357 The <Condition> element is a Boolean function over **attributes** or functions of **attributes**.

```
2358 <xs:element name="Condition" type="xacml:ConditionType"/>  
2359 <xs:complexType name="ConditionType">  
2360 <xs:sequence>  
2361 <xs:element ref="xacml:Expression"/>  
2362 </xs:sequence>  
2363 </xs:complexType>
```

2364 The <Condition> contains one <Expression> element, with the restriction that the <Expression>  
2365 return data-type MUST be "http://www.w3.org/2001/XMLSchema#boolean". Evaluation of the  
2366 <Condition> element is described in Section 7.9.

## 2367 5.27 Element <Apply>

2368 The <Apply> element denotes application of a function to its arguments, thus encoding a function call.

2369 The <Apply> element can be applied to any combination of the members of the <Expression>  
2370 element substitution group. See Section 5.25.



```

2371 <xs:element name="Apply" type="xacml:ApplyType"
2372 substitutionGroup="xacml:Expression"/>
2373 <xs:complexType name="ApplyType">
2374   <xs:complexContent>
2375     <xs:extension base="xacml:ExpressionType">
2376       <xs:sequence>
2377         <xs:element ref="xacml:Description" minOccurs="0"/>
2378         <xs:element ref="xacml:Expression" minOccurs="0"
2379           maxOccurs="unbounded"/>
2380       </xs:sequence>
2381       <xs:attribute name="FunctionId" type="xs:anyURI"
2382         use="required"/>
2383     </xs:extension>
2384   </xs:complexContent>
2385 </xs:complexType>

```

2386 The <Apply> element is of ApplyType complex type.

2387 The <Apply> element contains the following attributes and elements:

2388 FunctionId [Required]

2389       The identifier of the function to be applied to the arguments. XACML-defined functions are  
2390       described in Appendix A.3.

2391 <Description> [Optional]

2392       A free-form description of the <Apply> element.

2393 <Expression> [Optional]

2394       Arguments to the function, which may include other functions.

## 2395 5.28 Element <Function>

2396 The <Function> element SHALL be used to name a function as an argument to the function defined by  
2397 the parent <Apply> element.

```

2398 <xs:element name="Function" type="xacml:FunctionType"
2399 substitutionGroup="xacml:Expression"/>
2400 <xs:complexType name="FunctionType">
2401   <xs:complexContent>
2402     <xs:extension base="xacml:ExpressionType">
2403       <xs:attribute name="FunctionId" type="xs:anyURI"
2404         use="required"/>
2405     </xs:extension>
2406   </xs:complexContent>
2407 </xs:complexType>

```

2408 The <Function> element is of FunctionType complex type.

2409 The <Function> element contains the following attribute:

2410 FunctionId [Required]

2411       The identifier of the function.

## 2412 5.29 Element <AttributeDesignator>

2413 The <AttributeDesignator> element retrieves a **bag** of values for a **named attribute** from the  
2414 request **context**. A **named attribute** SHALL be considered present if there is at least one **attribute** that  
2415 matches the criteria set out below.

2416 The <AttributeDesignator> element SHALL return a **bag** containing all the **attribute** values that are  
2417 matched by the **named attribute**. In the event that no matching **attribute** is present in the **context**, the

2418 MustBePresent attribute governs whether this element returns an empty **bag** or "Indeterminate". See  
2419 Section 7.3.5.

2420 The <AttributeDesignator> MAY appear in the <Match> element and MAY be passed to the  
2421 <Apply> element as an argument.

2422 The <AttributeDesignator> element is of the AttributeDesignatorType complex type.

```
2423 <xs:element name="AttributeDesignator" type="xacml:AttributeDesignatorType"  
2424 substitutionGroup="xacml:Expression"/>  
2425 <xs:complexType name="AttributeDesignatorType">  
2426   <xs:complexContent>  
2427     <xs:extension base="xacml:ExpressionType">  
2428       <xs:attribute name="Category" type="xs:anyURI"  
2429         use="required"/>  
2430       <xs:attribute name="AttributeId" type="xs:anyURI"  
2431         use="required"/>  
2432       <xs:attribute name="DataType" type="xs:anyURI"  
2433         use="required"/>  
2434       <xs:attribute name="Issuer" type="xs:string" use="optional"/>  
2435       <xs:attribute name="MustBePresent" type="xs:boolean"  
2436         use="required"/>  
2437     </xs:extension>  
2438   </xs:complexContent>  
2439 </xs:complexType>
```

2440 A **named attribute** SHALL match an **attribute** if the values of their respective Category,  
2441 AttributeId, DataType and Issuer attributes match. The attribute designator's Category MUST  
2442 match, by **identifier equality**, the Category of the <Attributes> element in which the **attribute** is  
2443 present. The attribute designator's AttributeId MUST match, by **identifier equality**, the  
2444 AttributeId of the attribute. The attribute designator's DataType MUST match, by **identifier**  
2445 **equality**, the DataType of the same **attribute**.

2446 If the Issuer attribute is present in the attribute designator, then it MUST match, using the  
2447 "urn:oasis:names:tc:xacml:1.0:function:string-equal" function, the Issuer of the same **attribute**. If the  
2448 Issuer is not present in the attribute designator, then the matching of the **attribute** to the **named**  
2449 **attribute** SHALL be governed by AttributeId and DataType attributes alone.

2450 The <AttributeDesignatorType> contains the following attributes:

2451 Category [Required]

2452 This attribute SHALL specify the Category with which to match the **attribute**.

2453 AttributeId [Required]

2454 This attribute SHALL specify the AttributeId with which to match the **attribute**.

2455 DataType [Required]

2456 The **bag** returned by the <AttributeDesignator> element SHALL contain values of this data-  
2457 type.

2458 Issuer [Optional]

2459 This attribute, if supplied, SHALL specify the Issuer with which to match the **attribute**.

2460 MustBePresent [Required]

2461 This attribute governs whether the element returns "Indeterminate" or an empty **bag** in the event  
2462 the **named attribute** is absent from the request **context**. See Section 7.3.5. Also see Sections  
2463 7.19.2 and 7.19.3.

2464 **5.30 Element <AttributeSelector>**

2465 The <AttributeSelector> element produces a **bag** of unnamed and uncategorized **attribute** values.  
2466 The values shall be constructed from the node(s) selected by applying the XPath expression given by the  
2467 element's Path attribute to the XML content indicated by the element's Category attribute. Support for  
2468 the <AttributeSelector> element is OPTIONAL.

2469 See section 7.3.7 for details of <AttributeSelector> evaluation.

```
2470 <xs:element name="AttributeSelector" type="xacml:AttributeSelectorType"  
2471 substitutionGroup="xacml:Expression"/>  
2472 <xs:complexType name="AttributeSelectorType">  
2473 <xs:complexContent>  
2474 <xs:extension base="xacml:ExpressionType">  
2475 <xs:attribute name="Category" type="xs:anyURI"  
2476 use="required"/>  
2477 <xs:attribute name="ContextSelectorId" type="xs:anyURI"  
2478 use="optional"/>  
2479 <xs:attribute name="Path" type="xs:string"  
2480 use="required"/>  
2481 <xs:attribute name="DataType" type="xs:anyURI"  
2482 use="required"/>  
2483 <xs:attribute name="MustBePresent" type="xs:boolean"  
2484 use="required"/>  
2485 </xs:extension>  
2486 </xs:complexContent>  
2487 </xs:complexType>
```

2488 The <AttributeSelector> element is of AttributeSelectorType complex type.

2489 The <AttributeSelector> element has the following attributes:

2490 Category [Required]

2491 This attribute SHALL specify the **attributes** category of the <Content> element containing the  
2492 XML from which nodes will be selected. It also indicates the **attributes** category containing the  
2493 applicable ContextSelectorId attribute, if the element includes a ContextSelectorId xml  
2494 attribute.

2495 ContextSelectorId [Optional]

2496 This attribute refers to the **attribute** (by its AttributeId) in the request **context** in the category  
2497 given by the Category attribute. The referenced **attribute** MUST have data type  
2498 urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression, and must select a single node in the  
2499 <Content> element. The XPathCategory attribute of the referenced **attribute** MUST be equal  
2500 to the Category attribute of the **attribute selector**.

2501 Path [Required]

2502 This attribute SHALL contain an XPath expression to be evaluated against the specified XML  
2503 content. See Section 7.3.7 for details of the XPath evaluation during <AttributeSelector>  
2504 processing. The namespace context for the value of the Path attribute is given by the [in-scope  
2505 namespaces] [INFOSET] of the <AttributeSelector> element.

Formatted: Normal, Indent: Left: 0.5"

2506 DataType [Required]

2507 The attribute specifies the datatype of the values returned from the evaluation of this  
2508 <AttributeSelector> element.

2509 MustBePresent [Required]

2510 This attribute governs whether the element returns "Indeterminate" or an empty **bag** in the event  
2511 that the attributes category specified by the Category attribute does not exist in the request  
2512 context, or the attributes category does exist but it does not have a <Content> child element, or

Formatted: Normal, Indent: Left: 0.5"

Deleted: the XPath expression selects no node.

2514 [the <Content> element does exist but the XPath expression selects no node.](#) See Section 7.3.5.  
2515 Also see Sections 7.19.2 and 7.19.3.

### 2516 5.31 Element <AttributeValue>

2517 The <AttributeValue> element SHALL contain a literal *attribute* value.

```
2518 <xs:element name="AttributeValue" type="xacml:AttributeValueType"  
2519 substitutionGroup="xacml:Expression"/>  
2520 <xs:complexType name="AttributeValueType" mixed="true">  
2521 <xs:complexContent mixed="true">  
2522 <xs:extension base="xacml:ExpressionType">  
2523 <xs:sequence>  
2524 <xs:any namespace="##any" processContents="lax"  
2525 minOccurs="0" maxOccurs="unbounded"/>  
2526 </xs:sequence>  
2527 <xs:attribute name="DataType" type="xs:anyURI"  
2528 use="required"/>  
2529 <xs:anyAttribute namespace="##any" processContents="lax"/>  
2530 </xs:extension>  
2531 </xs:complexContent>  
2532 </xs:complexType>
```

2533 The <AttributeValue> element is of AttributeValueType complex type.

2534 The <AttributeValue> element has the following attributes:

2535 DataType [Required]

2536 The data-type of the *attribute* value.

### 2537 5.32 Element <Obligations>

2538 The <Obligations> element SHALL contain a set of <Obligation> elements.

```
2539 <xs:element name="Obligations" type="xacml:ObligationsType"/>  
2540 <xs:complexType name="ObligationsType">  
2541 <xs:sequence>  
2542 <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>  
2543 </xs:sequence>  
2544 </xs:complexType>
```

2545 The <Obligations> element is of ObligationsType complexType.

2546 The <Obligations> element contains the following element:

2547 <Obligation> [One to Many]

2548 A sequence of *obligations*. See Section 5.34.

### 2549 5.33 Element <AssociatedAdvice>

2550 The <AssociatedAdvice> element SHALL contain a set of <Advice> elements.

```
2551 <xs:element name="AssociatedAdvice" type="xacml:AssociatedAdviceType"/>  
2552 <xs:complexType name="AssociatedAdviceType">  
2553 <xs:sequence>  
2554 <xs:element ref="xacml:Advice" maxOccurs="unbounded"/>  
2555 </xs:sequence>  
2556 </xs:complexType>
```

2557 The <AssociatedAdvice> element is of AssociatedAdviceType complexType.

2558 The <AssociatedAdvice> element contains the following element:

2559 <Advice> [One to Many]

2560 A sequence of **advice**. See Section 5.35.

### 2561 5.34 Element <Obligation>

2562 The <Obligation> element SHALL contain an identifier for the **obligation** and a set of **attributes** that  
2563 form arguments of the action defined by the **obligation**.

```
2564 <xs:element name="Obligation" type="xacml:ObligationType"/>
2565 <xs:complexType name="ObligationType">
2566   <xs:sequence>
2567     <xs:element ref="xacml:AttributeAssignment" minOccurs="0"
2568       maxOccurs="unbounded"/>
2569   </xs:sequence>
2570   <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
2571 </xs:complexType>
```

2572 The <Obligation> element is of ObligationType complexType. See Section 7.18 for a description  
2573 of how the set of **obligations** to be returned by the **PDP** is determined.

2574 The <Obligation> element contains the following elements and attributes:

2575 ObligationId [Required]

2576 **Obligation** identifier. The value of the **obligation** identifier SHALL be interpreted by the **PEP**.

2577 <AttributeAssignment> [Optional]

2578 **Obligation** arguments assignment. The values of the **obligation** arguments SHALL be  
2579 interpreted by the **PEP**.

### 2580 5.35 Element <Advice>

2581 The <Advice> element SHALL contain an identifier for the **advice** and a set of **attributes** that form  
2582 arguments of the supplemental information defined by the **advice**.

```
2583 <xs:element name="Advice" type="xacml:AdviceType"/>
2584 <xs:complexType name="AdviceType">
2585   <xs:sequence>
2586     <xs:element ref="xacml:AttributeAssignment" minOccurs="0"
2587       maxOccurs="unbounded"/>
2588   </xs:sequence>
2589   <xs:attribute name="AdviceId" type="xs:anyURI" use="required"/>
2590 </xs:complexType>
```

2591 The <Advice> element is of AdviceType complexType. See Section 7.18 for a description of how the  
2592 set of **advice** to be returned by the **PDP** is determined.

2593 The <Advice> element contains the following elements and attributes:

2594 AdviceId [Required]

2595 **Advice** identifier. The value of the **advice** identifier MAY be interpreted by the **PEP**.

2596 <AttributeAssignment> [Optional]

2597 **Advice** arguments assignment. The values of the **advice** arguments MAY be interpreted by the  
2598 **PEP**.

### 2599 5.36 Element <AttributeAssignment>

2600 The <AttributeAssignment> element is used for including arguments in **obligation** and **advice**  
2601 expressions. It SHALL contain an **AttributeId** and the corresponding **attribute** value, by extending  
2602 the **AttributeValueType** type definition. The <AttributeAssignment> element MAY be used in  
2603 any way that is consistent with the schema syntax, which is a sequence of <xs:any> elements. The

2604 value specified SHALL be understood by the *PEP*, but it is not further specified by XACML. See Section  
2605 7.18. Section 4.2.4.3 provides a number of examples of arguments included in *obligation* expressions.

```
2606 <xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
2607 <xs:complexType name="AttributeAssignmentType" mixed="true">
2608   <xs:complexContent>
2609     <xs:extension base="xacml:AttributeValueType">
2610       <xs:attribute name="AttributeId" type="xs:anyURI"
2611         use="required"/>
2612       <xs:attribute name="Category" type="xs:anyURI"
2613         use="optional"/>
2614       <xs:attribute name="Issuer" type="xs:string" use="optional"/>
2615     </xs:extension>
2616   </xs:complexContent>
2617 </xs:complexType>
```

2618 The <AttributeAssignment> element is of AttributeAssignmentType complex type.

2619 The <AttributeAssignment> element contains the following attributes:

2620 AttributeId [Required]

2621 The *attribute* Identifier.

2622 Category [Optional]

2623 An optional category of the *attribute*. If this attribute is missing, the *attribute* has no category.

2624 The *PEP* SHALL interpret the significance and meaning of any Category attribute. Non-  
2625 normative note: an expected use of the category is to disambiguate *attributes* which are relayed  
2626 from the request.

2627 Issuer [Optional]

2628 An optional issuer of the *attribute*. If this attribute is missing, the *attribute* has no issuer. The  
2629 *PEP* SHALL interpret the significance and meaning of any Issuer attribute. Non-normative note:  
2630 an expected use of the issuer is to disambiguate *attributes* which are relayed from the request.

### 2631 5.37 Element <ObligationExpressions>

2632 The <ObligationExpressions> element SHALL contain a set of <ObligationExpression>  
2633 elements.

```
2634 <xs:element name="ObligationExpressions"
2635   type="xacml:ObligationExpressionsType"/>
2636 <xs:complexType name="ObligationExpressionsType">
2637   <xs:sequence>
2638     <xs:element ref="xacml:ObligationExpression" maxOccurs="unbounded"/>
2639   </xs:sequence>
2640 </xs:complexType>
```

2641 The <ObligationExpressions> element is of ObligationExpressionsType complexType.

2642 The <ObligationExpressions> element contains the following element:

2643 <ObligationExpression> [One to Many]

2644 A sequence of *obligations* expressions. See Section 5.39.

### 2645 5.38 Element <AdviceExpressions>

2646 The <AdviceExpressions> element SHALL contain a set of <AdviceExpression> elements.

```
2647 <xs:element name="AdviceExpressions" type="xacml:AdviceExpressionsType"/>
2648 <xs:complexType name="AdviceExpressionsType">
2649   <xs:sequence>
2650     <xs:element ref="xacml:AdviceExpression" maxOccurs="unbounded"/>
2651   </xs:sequence>
```

2652 </xs:complexType>

2653 The <AdviceExpressions> element is of AdviceExpressionsType complexType.

2654 The <AdviceExpressions> element contains the following element:

2655 <AdviceExpression> [One to Many]

2656 A sequence of **advice** expressions. See Section 5.40.

### 2657 5.39 Element <ObligationExpression>

2658 The <ObligationExpression> element evaluates to an **obligation** and SHALL contain an identifier for an **obligation** and a set of expressions that form arguments of the action defined by the **obligation**.

2660 The FulfillOn attribute SHALL indicate the **effect** for which this **obligation** must be fulfilled by the **PEP**.

```
2662 <xs:element name="ObligationExpression"
2663   type="xacml:ObligationExpressionType"/>
2664 <xs:complexType name="ObligationExpressionType">
2665   <xs:sequence>
2666     <xs:element ref="xacml:AttributeAssignmentExpression" minOccurs="0"
2667       maxOccurs="unbounded"/>
2668   </xs:sequence>
2669   <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
2670   <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>
2671 </xs:complexType>
```

2672 The <ObligationExpression> element is of ObligationExpressionType complexType. See Section 7.18 for a description of how the set of **obligations** to be returned by the **PDP** is determined.

2674 The <ObligationExpression> element contains the following elements and attributes:

2675 ObligationId [Required]

2676 **Obligation** identifier. The value of the **obligation** identifier SHALL be interpreted by the **PEP**.

2677 FulfillOn [Required]

2678 The **effect** for which this **obligation** must be fulfilled by the **PEP**.

2679 <AttributeAssignmentExpression> [Optional]

2680 **Obligation** arguments in the form of expressions. The expressions SHALL be evaluated by the PDP to constant <AttributeValue> elements or **bags**, which shall be the attribute assignments in the <Obligation> returned to the PEP. If an

2683 <AttributeAssignmentExpression> evaluates to an atomic **attribute** value, then there

2684 MUST be one resulting <AttributeAssignment> which MUST contain this single **attribute**

2685 value. If the <AttributeAssignmentExpression> evaluates to a **bag**, then there MUST be a

2686 resulting <AttributeAssignment> for each of the values in the **bag**. If the **bag** is empty, there

2687 shall be no <AttributeAssignment> from this <AttributeAssignmentExpression>. The

2688 values of the **obligation** arguments SHALL be interpreted by the **PEP**.

### 2689 5.40 Element <AdviceExpression>

2690 The <AdviceExpression> element evaluates to an **advice** and SHALL contain an identifier for an

2691 **advice** and a set of expressions that form arguments of the supplemental information defined by the

2692 **advice**. The AppliesTo attribute SHALL indicate the **effect** for which this **advice** must be provided to

2693 the **PEP**.

```
2694 <xs:element name="AdviceExpression" type="xacml:AdviceExpressionType"/>
2695 <xs:complexType name="AdviceExpressionType">
2696   <xs:sequence>
2697     <xs:element ref="xacml:AttributeAssignmentExpression" minOccurs="0"
2698       maxOccurs="unbounded"/>
```

```

2699 </xs:sequence>
2700 <xs:attribute name="AdviceId" type="xs:anyURI" use="required"/>
2701 <xs:attribute name="AppliesTo" type="xacml:EffectType" use="required"/>
2702 </xs:complexType>

```

2703 The <AdviceExpression> element is of AdviceExpressionType complexType. See Section 7.18  
 2704 for a description of how the set of **advice** to be returned by the **PDP** is determined.

2705 The <AdviceExpression> element contains the following elements and attributes:

2706 AdviceId [Required]

2707 **Advice** identifier. The value of the **advice** identifier MAY be interpreted by the **PEP**.

2708 AppliesTo [Required]

2709 The **effect** for which this **advice** must be provided to the **PEP**.

2710 <AttributeAssignmentExpression> [Optional]

2711 **Advice** arguments in the form of expressions. The expressions SHALL be evaluated by the PDP  
 2712 to constant <AttributeValue> elements or **bags**, which shall be the attribute assignments in  
 2713 the <Advice> returned to the PEP. If an <AttributeAssignmentExpression> evaluates to  
 2714 an atomic **attribute** value, then there MUST be one resulting <AttributeAssignment> which  
 2715 MUST contain this single **attribute** value. If the <AttributeAssignmentExpression>  
 2716 evaluates to a **bag**, then there MUST be a resulting <AttributeAssignment> for each of the  
 2717 values in the **bag**. If the **bag** is empty, there shall be no <AttributeAssignment> from this  
 2718 <AttributeAssignmentExpression>. The values of the **advice** arguments MAY be  
 2719 interpreted by the **PEP**.

## 2720 5.41 Element <AttributeAssignmentExpression>

2721 The <AttributeAssignmentExpression> element is used for including arguments in **obligations**  
 2722 and **advice**. It SHALL contain an AttributeId and an expression which SHALL be evaluated into the  
 2723 corresponding **attribute** value. The value specified SHALL be understood by the **PEP**, but it is not further  
 2724 specified by XACML. See Section 7.18. Section 4.2.4.3 provides a number of examples of arguments  
 2725 included in **obligations**.

```

2726 <xs:element name="AttributeAssignmentExpression"
2727   type="xacml:AttributeAssignmentExpressionType"/>
2728 <xs:complexType name="AttributeAssignmentExpressionType">
2729   <xs:sequence>
2730     <xs:element ref="xacml:Expression"/>
2731   </xs:sequence>
2732   <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
2733   <xs:attribute name="Category" type="xs:anyURI" use="optional"/>
2734   <xs:attribute name="Issuer" type="xs:string" use="optional"/>
2735 </xs:complexType>

```

2736 The <AttributeAssignmentExpression> element is of AttributeAssignmentExpressionType  
 2737 complex type.

2738 The <AttributeAssignmentExpression> element contains the following attributes:

2739 <Expression> [Required]

2740 The expression which evaluates to a constant **attribute** value or a bag of zero or more attribute  
 2741 values. See section 5.25.

2742 AttributeId [Required]

2743 The **attribute** identifier. The value of the AttributeId attribute in the resulting  
 2744 <AttributeAssignment> element MUST be equal to this value.

2745 Category [Optional]



2746 An optional category of the **attribute**. If this attribute is missing, the **attribute** has no category.  
2747 The value of the `Category` attribute in the resulting `<AttributeAssignment>` element MUST be  
2748 equal to this value.

2749 Issuer [Optional]

2750 An optional issuer of the **attribute**. If this attribute is missing, the **attribute** has no issuer. The  
2751 value of the `Issuer` attribute in the resulting `<AttributeAssignment>` element MUST be equal to  
2752 this value.

## 2753 5.42 Element `<Request>`

2754 The `<Request>` element is an abstraction layer used by the **policy** language. For simplicity of  
2755 expression, this document describes **policy** evaluation in terms of operations on the **context**. However a  
2756 conforming **PDP** is not required to actually instantiate the **context** in the form of an XML document. But,  
2757 any system conforming to the XACML specification MUST produce exactly the same **authorization**  
2758 **decisions** as if all the inputs had been transformed into the form of an `<Request>` element.

```
2759 <xs:element name="Request" type="xacml:RequestType"/>  
2760 <xs:complexType name="RequestType">  
2761 <xs:sequence>  
2762 <xs:element ref="xacml:RequestDefaults" minOccurs="0"/>  
2763 <xs:element ref="xacml:Attributes" maxOccurs="unbounded"/>  
2764 <xs:element ref="xacml:MultiRequests" minOccurs="0"/>  
2765 </xs:sequence>  
2766 <xs:attribute name="ReturnPolicyIdList" type="xs:boolean" use="required"/>  
2767 <xs:attribute name="CombinedDecision" type="xs:boolean" use="required" />  
2768 </xs:complexType>
```

**Deleted:** The `<Request>` element contains `<Attributes>` elements. There may be multiple `<Attributes>` elements with the same `Category` attribute if the **PDP** implements the multiple decision profile, see **[Multi]**. Under other conditions, it is a syntax error if there are multiple `<Attributes>` elements with the same `Category` (see Section 7.19.2 for error codes).¶

**Moved down [1]:** for error codes).¶

2769 The `<Request>` element is of `RequestType` complex type.

2770 The `<Request>` element contains the following elements and attributes:

2771 `ReturnPolicyIdList` [Required]

2772 This attribute is used to request that the **PDP** return a list of all fully applicable **policies** and  
2773 **policy sets** which were used in the decision as a part of the decision response.

2774 `CombinedDecision` [Required]

2775 This attribute is used to request that the **PDP** combines multiple decisions into a single decision.  
2776 The use of this attribute is specified in **[Multi]**. If the **PDP** does not implement the relevant  
2777 functionality in **[Multi]**, then the **PDP** must return an Indeterminate with a status code of  
2778 `urn:oasis:names:tc:xacml:1.0:status:processing-error` if it receives a request with this attribute set  
2779 to "true".

2780 `<RequestDefaults>` [Optional]

2781 Contains default values for the request, such as XPath version. See section 5.43.

2782 `<Attributes>` [One to Many]

2783 Specifies information about **attributes** of the request **context** by listing a sequence of  
2784 `<Attribute>` elements associated with an **attribute** category. One or more `<Attributes>`  
2785 elements are allowed. Different `<Attributes>` elements with different categories are used to  
2786 represent information about the **subject**, **resource**, **action**, **environment** or other categories of  
2787 the **access** request.

2788 The `<Request>` element contains `<Attributes>` elements. There may be multiple  
2789 `<Attributes>` elements with the same `Category` attribute if the **PDP** implements the multiple  
2790 decision profile, see **[Multi]**. Under other conditions, it is a syntax error if there are multiple  
2791 `<Attributes>` elements with the same `Category` (see Section 7.19.2 for error codes).

**Moved (insertion) [1]**

**Formatted:** Definition

2792 `<MultiRequests>` [Optional]

2801 Lists multiple **request contexts** by references to the <Attributes> elements. Implementation  
2802 of this element is optional. The semantics of this element is defined in **[Multi]**. If the  
2803 implementation does not implement this element, it MUST return an Indeterminate result if it  
2804 encounters this element. See section 5.50.

#### 2805 5.43 Element <RequestDefaults>

2806 The <RequestDefaults> element SHALL specify default values that apply to the <Request> element.

```
2807 <xs:element name="RequestDefaults" type="xacml:RequestDefaultsType"/>  
2808 <xs:complexType name="RequestDefaultsType">  
2809   <xs:sequence>  
2810     <xs:choice>  
2811       <xs:element ref="xacml:XPathVersion"/>  
2812     </xs:choice>  
2813   </xs:sequence>  
2814 </xs:complexType>
```

2815 <RequestDefaults> element is of RequestDefaultsType complex type.

2816 The <RequestDefaults> element contains the following elements:

2817 <XPathVersion> [Optional]

2818 Default XPath version for XPath expressions occurring in the request.

#### 2819 5.44 Element <Attributes>

2820 The <Attributes> element specifies **attributes** of a **subject, resource, action, environment** or  
2821 another category by listing a sequence of <Attribute> elements associated with the category.

```
2822 <xs:element name="Attributes" type="xacml:AttributesType"/>  
2823 <xs:complexType name="AttributesType">  
2824   <xs:sequence>  
2825     <xs:element ref="xacml:Content" minOccurs="0"/>  
2826     <xs:element ref="xacml:Attribute" minOccurs="0"  
2827       maxOccurs="unbounded"/>  
2828   </xs:sequence>  
2829   <xs:attribute name="Category" type="xs:anyURI" use="required"/>  
2830   <xs:attribute ref="xml:id" use="optional"/>  
2831 </xs:complexType>
```

Deleted: <xs:complexType name="SubjectType">

2832 The <Attributes> element is of AttributesType complex type.

2833 The <Attributes> element contains the following elements and attributes:

2834 Category [Required]

2835 This attribute indicates which **attribute** category the contained **attributes** belong to. The  
2836 Category attribute is used to differentiate between **attributes** of **subject, resource, action,**  
2837 **environment** or other categories.

2838 xml:id [Optional]

2839 This attribute provides a unique identifier for this <Attributes> element. See **[XMLid]** It is  
2840 primarily intended to be referenced in multiple requests. See **[Multi]**.

2841 <Content> [Optional]

2842 Specifies additional sources of **attributes** in free form XML document format which can be  
2843 referenced using <AttributeSelector> elements.

2844 <Attribute> [Any Number]

2845 A sequence of **attributes** that apply to the category of the request.

## 2847 5.45 Element <Content>

2848 The <Content> element is a notional placeholder for additional **attributes**, typically the content of the  
2849 **resource**.

```
2850 <xs:element name="Content" type="xacml:ContentType"/>  
2851 <xs:complexType name="ContentType" mixed="true">  
2852   <xs:sequence>  
2853     <xs:any namespace="##any" processContents="lax"/>  
2854   </xs:sequence>  
2855 </xs:complexType>
```

2856 The <Content> element is of ContentType complex type.

2857 The <Content> element has exactly one arbitrary type child element.

## 2858 5.46 Element <Attribute>

2859 The <Attribute> element is the central abstraction of the request **context**. It contains **attribute** meta-  
2860 data and one or more **attribute** values. The **attribute** meta-data comprises the **attribute** identifier and  
2861 the **attribute** issuer. <AttributeDesignator> elements in the **policy** MAY refer to **attributes** by  
2862 means of this meta-data.

```
2863 <xs:element name="Attribute" type="xacml:AttributeType"/>  
2864 <xs:complexType name="AttributeType">  
2865   <xs:sequence>  
2866     <xs:element ref="xacml:AttributeValue" maxOccurs="unbounded"/>  
2867   </xs:sequence>  
2868   <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>  
2869   <xs:attribute name="Issuer" type="xs:string" use="optional"/>  
2870   <xs:attribute name="IncludeInResult" type="xs:boolean" use="required"/>  
2871 </xs:complexType>
```

2872 The <Attribute> element is of AttributeType complex type.

2873 The <Attribute> element contains the following attributes and elements:

2874 AttributeId [Required]

2875 The **Attribute** identifier. A number of identifiers are reserved by XACML to denote commonly  
2876 used **attributes**. See Appendix Appendix B.

2877 Issuer [Optional]

2878 The **Attribute** issuer. For example, this attribute value MAY be an x500Name that binds to a  
2879 public key, or it may be some other identifier exchanged out-of-band by issuing and relying  
2880 parties.

2881 IncludeInResult [Default: false]

2882 Whether to include this **attribute** in the result. This is useful to correlate requests with their  
2883 responses in case of multiple requests.

2884 <AttributeValue> [One to Many]

2885 One or more **attribute** values. Each **attribute** value MAY have contents that are empty, occur  
2886 once or occur multiple times.

## 2887 5.47 Element <Response>

2888 The <Response> element is an abstraction layer used by the **policy** language. Any proprietary system  
2889 using the XACML specification MUST transform an XACML **context** <Response> element into the form  
2890 of its **authorization decision**.

2891 The <Response> element encapsulates the **authorization decision** produced by the **PDP**. It includes a  
2892 sequence of one or more results, with one <Result> element per requested **resource**. Multiple results

2893 MAY be returned by some implementations, in particular those that support the XACML Profile for  
2894 Requests for Multiple Resources **[Multi]**. Support for multiple results is OPTIONAL.

```
2895 <xs:element name="Response" type="xacml:ResponseType"/>
2896 <xs:complexType name="ResponseType">
2897   <xs:sequence>
2898     <xs:element ref="xacml:Result" maxOccurs="unbounded"/>
2899   </xs:sequence>
2900 </xs:complexType>
```

2901 The <Response> element is of ResponseType complex type.

2902 The <Response> element contains the following elements:

2903 <Result> [One to Many]

2904 An **authorization decision** result. See Section 5.48.

## 2905 5.48 Element <Result>

2906 The <Result> element represents an **authorization decision** result. It MAY include a set of  
2907 **obligations** that MUST be fulfilled by the **PEP**. If the **PEP** does not understand or cannot fulfill an  
2908 **obligation**, then the action of the PEP is determined by its bias, see section 7.1. It MAY include a set of  
2909 **advice** with supplemental information which MAY be safely ignored by the **PEP**.

```
2910 <xs:complexType name="ResultType">
2911   <xs:sequence>
2912     <xs:element ref="xacml:Decision"/>
2913     <xs:element ref="xacml:Status" minOccurs="0"/>
2914     <xs:element ref="xacml:Obligations" minOccurs="0"/>
2915     <xs:element ref="xacml:AssociatedAdvice" minOccurs="0"/>
2916     <xs:element ref="xacml:Attributes" minOccurs="0"
2917       maxOccurs="unbounded"/>
2918     <xs:element ref="xacml:PolicyIdentifierList" minOccurs="0"/>
2919   </xs:sequence>
2920 </xs:complexType>
```

2921 The <Result> element is of ResultType complex type.

2922 The <Result> element contains the following attributes and elements:

2923 <Decision> [Required]

2924 The **authorization decision**: "Permit", "Deny", "Indeterminate" or "NotApplicable".

2925 <Status> [Optional]

2926 Indicates whether errors occurred during evaluation of the **decision request**, and optionally,  
2927 information about those errors. If the <Response> element contains <Result> elements whose  
2928 <Status> elements are all identical, and the <Response> element is contained in a protocol  
2929 wrapper that can convey status information, then the common status information MAY be placed  
2930 in the protocol wrapper and this <Status> element MAY be omitted from all <Result>  
2931 elements.

2932 <Obligations> [Optional]

2933 A list of **obligations** that MUST be fulfilled by the **PEP**. If the **PEP** does not understand or cannot  
2934 fulfill an **obligation**, then the action of the PEP is determined by its bias, see section 7.2. See  
2935 Section 7.18 for a description of how the set of **obligations** to be returned by the **PDP** is  
2936 determined.

2937 <AssociatedAdvice> [Optional]

2938 A list of **advice** that provide supplemental information to the **PEP**. If the **PEP** does not  
2939 understand an **advice**, the PEP may safely ignore the **advice**. See Section 7.18 for a description  
2940 of how the set of **advice** to be returned by the **PDP** is determined.

2941 <Attributes> [Optional]

2942 A list of **attributes** that were part of the request. The choice of which **attributes** are included here  
2943 is made with the IncludeInResult attribute of the <Attribute> elements of the request. See  
2944 section 5.46.

2945 <PolicyIdentifierList> [Optional]

2946 If the ReturnPolicyIdList attribute in the <Request> is true (see section 5.42), a **PDP** that  
2947 implements this optional feature **MUST** return a list which includes the identifiers of all policies  
2948 which were found to be fully applicable, whether or not the <Effect> was the same or different  
2949 from the <Decision>. The list MAY include the identifiers of other policies which are currently in  
2950 force, as long as no policies required for the decision are omitted. A PDP MAY satisfy this  
2951 requirement by including all policies currently in force, or by including all policies which were  
2952 evaluated in making the decision, or by including all policies which did not evaluate to  
2953 "NotApplicable", or by any other algorithm which does not omit any policies which contributed to  
2954 the decision. However, a decision which returns "NotApplicable" MUST return an empty list.  
2955

Deleted: 5.42),

Formatted: Glossary term

Deleted: . That is, all **policies** where both the <Target> matched and the <Condition> evaluated to true

## 2956 5.49 Element <PolicyIdentifierList>

2957 The <PolicyIdentifierList> element contains a list of **policy** and **policy set** identifiers of **policies**  
2958 which have been applicable to a request. The list is unordered.

```
2959 <xs:element name="PolicyIdentifierList"  
2960 type="xacml:PolicyIdentifierListType"/>  
2961 <xs:complexType name="PolicyIdentifierListType">  
2962 <xs:choice minOccurs="0" maxOccurs="unbounded">  
2963 <xs:element ref="xacml:PolicyIdReference"/>  
2964 <xs:element ref="xacml:PolicySetIdReference"/>  
2965 </xs:choice>  
2966 </xs:complexType>
```

2967 The <PolicyIdentifierList> element is of PolicyIdentifierListType complex type.

2968 The <PolicyIdentifierList> element contains the following elements.

2969 <PolicyIdReference> [Any number]

2970 The identifier and version of a **policy** which was applicable to the request. See section 5.11. The  
2971 <PolicyIdReference> element **MUST** use the Version attribute to specify the version and  
2972 **MUST NOT** use the LatestVersion or EarliestVersion attributes.

2973 <PolicySetIdReference> [Any number]

2974 The identifier and version of a **policy set** which was applicable to the request. See section 5.10.  
2975 The <PolicySetIdReference> element **MUST** use the Version attribute to specify the  
2976 version and **MUST NOT** use the LatestVersion or EarliestVersion attributes.

## 2977 5.50 Element <MultiRequests>

2978 The <MultiRequests> element contains a list of requests by reference to <Attributes> elements in  
2979 the enclosing <Request> element. The semantics of this element are defined in [Multi]. Support for this  
2980 element is optional. If an implementation does not support this element, but receives it, the  
2981 implementation **MUST** generate an "Indeterminate" response.

```
2982 <xs:element name="MultiRequests" type="xacml:MultiRequestsType"/>  
2983 <xs:complexType name="MultiRequestsType">  
2984 <xs:sequence>  
2985 <xs:element ref="xacml:RequestReference" maxOccurs="unbounded"/>  
2986 </xs:sequence>  
2987 </xs:complexType>
```

2991 The <MultiRequests> element contains the following elements.

2992 <RequestReference> [one to many]

2993 Defines a request instance by reference to <Attributes> elements in the enclosing  
2994 <Request> element. See section 5.51.

## 2995 5.51 Element <RequestReference>

2996 The <RequestReference> element defines an instance of a request in terms of references to  
2997 <Attributes> elements. The semantics of this element are defined in [Multi]. Support for this element  
2998 is optional.

```
2999 <xs:element name="RequestReference" type="xacml:RequestReference" />  
3000 <xs:complexType name="RequestReferenceType">  
3001   <xs:sequence>  
3002     <xs:element ref="xacml:AttributesReference" maxOccurs="unbounded"/>  
3003   </xs:sequence>  
3004 </xs:complexType>
```

3005 The <RequestReference> element contains the following elements.

3006 <AttributesReference> [one to many]

3007 A reference to an <Attributes> element in the enclosing <Request> element. See section  
3008 5.52.

## 3009 5.52 Element <AttributesReference>

3010 The <AttributesReference> element makes a reference to an <Attributes> element. The  
3011 meaning of this element is defined in [Multi]. Support for this element is optional.

```
3012 <xs:element name="AttributesReference" type="xacml:AttributesReference" />  
3013 <xs:complexType name="AttributesReferenceType">  
3014   <xs:attribute name="ReferenceId" type="xs:IDREF" use="required" />  
3015 </xs:complexType>
```

3016 The <AttributesReference> element contains the following attributes.

3017 ReferenceId [required]

3018 A reference to the xml:id attribute of an <Attributes> element in the enclosing <Request>  
3019 element.

## 3020 5.53 Element <Decision>

3021 The <Decision> element contains the result of *policy* evaluation.

```
3022 <xs:element name="Decision" type="xacml:DecisionType" />  
3023 <xs:simpleType name="DecisionType">  
3024   <xs:restriction base="xs:string">  
3025     <xs:enumeration value="Permit"/>  
3026     <xs:enumeration value="Deny"/>  
3027     <xs:enumeration value="Indeterminate"/>  
3028     <xs:enumeration value="NotApplicable"/>  
3029   </xs:restriction>  
3030 </xs:simpleType>
```

3031 The <Decision> element is of DecisionType simple type.

3032 The values of the <Decision> element have the following meanings:

3033 "Permit": the requested **access** is permitted.

3034 "Deny": the requested **access** is denied.

3035 "Indeterminate": the **PDP** is unable to evaluate the requested **access**. Reasons for such inability  
3036 include: missing **attributes**, network errors while retrieving **policies**, division by zero during  
3037 **policy** evaluation, syntax errors in the **decision request** or in the **policy**, etc.  
3038 "NotApplicable": the **PDP** does not have any **policy** that applies to this **decision request**.

## 3039 5.54 Element <Status>

3040 The <Status> element represents the status of the **authorization decision** result.

```
3041 <xs:element name="Status" type="xacml:StatusType"/>  
3042 <xs:complexType name="StatusType">  
3043   <xs:sequence>  
3044     <xs:element ref="xacml:StatusCode"/>  
3045     <xs:element ref="xacml:StatusMessage" minOccurs="0"/>  
3046     <xs:element ref="xacml:StatusDetail" minOccurs="0"/>  
3047   </xs:sequence>  
3048 </xs:complexType>
```

3049 The <Status> element is of StatusType complex type.

3050 The <Status> element contains the following elements:

3051 <StatusCode> [Required]

3052 Status code.

3053 <StatusMessage> [Optional]

3054 A status message describing the status code.

3055 <StatusDetail> [Optional]

3056 Additional status information.

## 3057 5.55 Element <StatusCode>

3058 The <StatusCode> element contains a major status code value and an optional **recursive series** of  
3059 minor status codes.

Deleted: sequence

```
3060 <xs:element name="StatusCode" type="xacml:StatusCodeType"/>  
3061 <xs:complexType name="StatusCodeType">  
3062   <xs:sequence>  
3063     <xs:element ref="xacml:StatusCode" minOccurs="0"/>  
3064   </xs:sequence>  
3065   <xs:attribute name="Value" type="xs:anyURI" use="required"/>  
3066 </xs:complexType>
```

3067 The <StatusCode> element is of StatusCodeType complex type.

3068 The <StatusCode> element contains the following attributes and elements:

3069 Value [Required]

3070 See Section B.8 for a list of values.

3071 <StatusCode> [Any Number]

3072 Minor status code. This status code qualifies its parent status code.

## 3073 5.56 Element <StatusMessage>

3074 The <StatusMessage> element is a free-form description of the status code.

```
3075 <xs:element name="StatusMessage" type="xs:string"/>
```

3076 The <StatusMessage> element is of xs:string type.

## 3078 5.57 Element <StatusDetail>

3079 The <StatusDetail> element qualifies the <Status> element with additional information.

```
3080 <xs:element name="StatusDetail" type="xacml:StatusDetailType"/>
3081 <xs:complexType name="StatusDetailType">
3082   <xs:sequence>
3083     <xs:any namespace="##any" processContents="lax" minOccurs="0"
3084       maxOccurs="unbounded"/>
3085   </xs:sequence>
3086 </xs:complexType>
```

3087 The <StatusDetail> element is of StatusDetailType complex type.

3088 The <StatusDetail> element allows arbitrary XML content.

3089 Inclusion of a <StatusDetail> element is optional. However, if a **PDP** returns one of the following  
3090 XACML-defined <StatusCode> values and includes a <StatusDetail> element, then the following  
3091 rules apply.

3092 urn:oasis:names:tc:xacml:1.0:status:ok

3093 A **PDP** MUST NOT return a <StatusDetail> element in conjunction with the “ok” status value.

3094 urn:oasis:names:tc:xacml:1.0:status:missing-attribute

3095 A **PDP** MAY choose not to return any <StatusDetail> information or MAY choose to return a  
3096 <StatusDetail> element containing one or more <MissingAttributeDetail> elements.

3097 urn:oasis:names:tc:xacml:1.0:status:syntax-error

3098 A **PDP** MUST NOT return a <StatusDetail> element in conjunction with the “syntax-error” status  
3099 value. A syntax error may represent either a problem with the **policy** being used or with the request  
3100 **context**. The **PDP** MAY return a <StatusMessage> describing the problem.

3101 urn:oasis:names:tc:xacml:1.0:status:processing-error

3102 A **PDP** MUST NOT return <StatusDetail> element in conjunction with the “processing-error” status  
3103 value. This status code indicates an internal problem in the **PDP**. For security reasons, the **PDP** MAY  
3104 choose to return no further information to the **PEP**. In the case of a divide-by-zero error or other  
3105 computational error, the **PDP** MAY return a <StatusMessage> describing the nature of the error.

## 3106 5.58 Element <MissingAttributeDetail>

3107 The <MissingAttributeDetail> element conveys information about **attributes** required for **policy**  
3108 evaluation that were missing from the request **context**.

```
3109 <xs:element name="MissingAttributeDetail"
3110   type="xacml:MissingAttributeDetailType"/>
3111 <xs:complexType name="MissingAttributeDetailType">
3112   <xs:sequence>
3113     <xs:element ref="xacml:AttributeValue" minOccurs="0"
3114       maxOccurs="unbounded"/>
3115   </xs:sequence>
3116   <xs:attribute name="Category" type="xs:anyURI" use="required"/>
3117   <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
3118   <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
3119   <xs:attribute name="Issuer" type="xs:string" use="optional"/>
3120 </xs:complexType>
```

3121 The <MissingAttributeDetail> element is of MissingAttributeDetailType complex type.

3122 The <MissingAttributeDetail> element contains the following attributes and elements:

3123 <AttributeValue> [Optional]

3124 The required value of the missing **attribute**.



- 3125 `Category` [Required]  
3126       The category identifier of the missing **attribute**.
- 3127 `AttributeId` [Required]  
3128       The identifier of the missing **attribute**.
- 3129 `DataType` [Required]  
3130       The data-type of the missing **attribute**.
- 3131 `Issuer` [Optional]  
3132       This attribute, if supplied, SHALL specify the required `Issuer` of the missing **attribute**.
- 3133 If the **PDP** includes `<AttributeValue>` elements in the `<MissingAttributeDetail>` element, then  
3134 this indicates the acceptable values for that **attribute**. If no `<AttributeValue>` elements are included,  
3135 then this indicates the names of **attributes** that the **PDP** failed to resolve during its evaluation. The list of  
3136 **attributes** may be partial or complete. There is no guarantee by the **PDP** that supplying the missing  
3137 values or **attributes** will be sufficient to satisfy the **policy**.

## 3138 6 XPath 2.0 definitions

3139 The XPath 2.0 specification leaves a number of aspects of behavior implementation defined. This section  
3140 defines how XPath 2.0 SHALL behave when hosted in XACML.

3141 <http://www.w3.org/TR/2007/REC-xpath20-20070123/#id-impl-defined-items> defines the following items:

- 3142 1. The version of Unicode that is used to construct expressions.  
3143 XACML leaves this implementation defined. It is RECOMMENDED that the latest version is used.
- 3144 2. The statically-known collations.  
3145 XACML leaves this implementation defined.
- 3146 3. The implicit timezone.  
3147 XACML defined the implicit time zone as UTC.
- 3148 4. The circumstances in which warnings are raised, and the ways in which warnings are handled.  
3149 XACML leaves this implementation defined.
- 3150 5. The method by which errors are reported to the external processing environment.  
3151 An XPath error causes an XACML Indeterminate value in the element where the XPath error  
3152 occurs. The StatusCode value SHALL be "urn:oasis:names:tc:xacml:1.0:status:processing-error".  
3153 Implementations MAY provide additional details about the error in the response or by some other  
3154 means.
- 3155 6. Whether the implementation is based on the rules of XML 1.0 or 1.1.  
3156 XACML is based on XML 1.0.
- 3157 7. Whether the implementation supports the namespace axis.  
3158 XACML leaves this implementation defined. It is RECOMMENDED that users of XACML do not  
3159 make use of the namespace axis.
- 3160 8. Any static typing extensions supported by the implementation, if the Static Typing Feature is  
3161 supported.  
3162 XACML leaves this implementation defined.

3163  
3164 <http://www.w3.org/TR/2007/REC-xpath-datamodel-20070123/#implementation-defined> defines the  
3165 following items:

- 3166 1. Support for additional user-defined or implementation-defined types is implementation-defined.  
3167 It is RECOMMENDED that implementations of XACML do not define any additional types and it is  
3168 RECOMMENDED that users of XACML do not make user of any additional types.
- 3169 2. Some typed values in the data model are undefined. Attempting to access an undefined property  
3170 is always an error. Behavior in these cases is implementation-defined and the host language is  
3171 responsible for determining the result.  
3172 An XPath error causes an XACML Indeterminate value in the element where the XPath error  
3173 occurs. The StatusCode value SHALL be "urn:oasis:names:tc:xacml:1.0:status:processing-error".  
3174 Implementations MAY provide additional details about the error in the response or by some other  
3175 means.

3176  
3177 <http://www.w3.org/TR/2007/REC-xpath-functions-20070123/#impl-def> defines the following items:

- 3178 1. The destination of the trace output is implementation-defined.  
3179 XACML leaves this implementation defined.
- 3180 2. For xs:integer operations, implementations that support limited-precision integer operations must  
3181 either raise an error [err:FOAR0002] or provide an implementation-defined mechanism that  
3182 allows users to choose between raising an error and returning a result that is modulo the largest  
3183 representable integer value.  
3184 XACML leaves this implementation defined. If an implementation chooses to raise an error, the

- 3185            StatusCode value SHALL be "urn:oasis:names:tc:xacml:1.0:status:processing-error".  
3186            Implementations MAY provide additional details about the error in the response or by some other  
3187            means.
- 3188            3. For xs:decimal values the number of digits of precision returned by the numeric operators is  
3189            implementation-defined.  
3190            XACML leaves this implementation defined.
- 3191            4. If the number of digits in the result of a numeric operation exceeds the number of digits that the  
3192            implementation supports, the result is truncated or rounded in an implementation-defined manner.  
3193            XACML leaves this implementation defined.
- 3194            5. It is implementation-defined which version of Unicode is supported.  
3195            XACML leaves this implementation defined. It is RECOMMENDED that the latest version is used.
- 3196            6. For fn:normalize-unicode, conforming implementations must support normalization form "NFC"  
3197            and may support normalization forms "NFD", "NFKC", "NFKD", "FULLY-NORMALIZED". They  
3198            may also support other normalization forms with implementation-defined semantics.  
3199            XACML leaves this implementation defined.
- 3200            7. The ability to decompose strings into collation units suitable for substring matching is an  
3201            implementation-defined property of a collation.  
3202            XACML leaves this implementation defined.
- 3203            8. All minimally conforming processors must support year values with a minimum of 4 digits (i.e.,  
3204            YYYY) and a minimum fractional second precision of 1 millisecond or three digits (i.e., s.sss).  
3205            However, conforming processors may set larger implementation-defined limits on the maximum  
3206            number of digits they support in these two situations.  
3207            XACML leaves this implementation defined, and it is RECOMMENDED that users of XACML do  
3208            not expect greater limits and precision.
- 3209            9. The result of casting a string to xs:decimal, when the resulting value is not too large or too small  
3210            but nevertheless has too many decimal digits to be accurately represented, is implementation-  
3211            defined.  
3212            XACML leaves this implementation defined.
- 3213            10. Various aspects of the processing provided by fn:doc are implementation-defined.  
3214            Implementations may provide external configuration options that allow any aspect of the  
3215            processing to be controlled by the user.  
3216            XACML leaves this implementation defined.
- 3217            11. The manner in which implementations provide options to weaken the stable characteristic of  
3218            fn:collection and fn:doc are implementation-defined.  
3219            XACML leaves this implementation defined.

---

## 3220 7 Functional requirements

3221 This section specifies certain functional requirements that are not directly associated with the production  
3222 or consumption of a particular XACML element.

3223 Note that in each case an implementation is conformant as long as it produces the same result as is  
3224 specified here, regardless of how and in what order the implementation behaves internally.

### 3225 7.1 Unicode issues

#### 3226 7.1.1 Normalization

3227 In Unicode, some equivalent characters can be represented by more than one different Unicode  
3228 character sequence. See [CMF]. The process of converting Unicode strings into equivalent character  
3229 sequences is called "normalization" [UAX15]. Some operations, such as string comparison, are sensitive  
3230 to normalization. An operation is normalization-sensitive if its output(s) are different depending on the  
3231 state of normalization of the input(s); if the output(s) are textual, they are deemed different only if they  
3232 would remain different were they to be normalized.

3233 For more information on normalization see [CM].

3234 An XACML implementation **MUST** behave as if each normalization-sensitive operation normalizes input  
3235 strings into Unicode Normalization Form C ("NFC"). An implementation **MAY** use some other form of  
3236 internal processing (such as using a non-Unicode, "legacy" character encoding) as long as the externally  
3237 visible results are identical to this specification.

#### 3238 7.1.2 Version of Unicode

3239 The version of Unicode used by XACML is implementation defined. It is RECOMMENDED that the latest  
3240 version is used. Also note security issues in section 9.3.

### 3241 7.2 Policy enforcement point

3242 This section describes the requirements for the *PEP*.

3243 An application functions in the role of the *PEP* if it guards *access* to a set of *resources* and asks the  
3244 *PDP* for an *authorization decision*. The *PEP* **MUST** abide by the *authorization decision* as described  
3245 in one of the following sub-sections

3246 In any case any *advice* in the *decision* may be safely ignored by the *PEP*.

#### 3247 7.2.1 Base PEP

3248 If the *decision* is "Permit", then the *PEP* **SHALL** permit *access*. If *obligations* accompany the *decision*,  
3249 then the *PEP* **SHALL** permit *access* only if it understands and it can and will discharge those  
3250 *obligations*.

3251 If the *decision* is "Deny", then the *PEP* **SHALL** deny *access*. If *obligations* accompany the *decision*,  
3252 then the *PEP* shall deny *access* only if it understands, and it can and will discharge those *obligations*.

3253 If the *decision* is "Not Applicable", then the *PEP*'s behavior is undefined.

3254 If the *decision* is "Indeterminate", then the *PEP*'s behavior is undefined.

#### 3255 7.2.2 Deny-biased PEP

3256 If the *decision* is "Permit", then the *PEP* **SHALL** permit *access*. If *obligations* accompany the *decision*,  
3257 then the *PEP* **SHALL** permit *access* only if it understands and it can and will discharge those  
3258 *obligations*.

3259 All other **decisions** SHALL result in the denial of **access**.

3260 Note: other actions, e.g. consultation of additional **PDPs**, reformulation/resubmission of  
3261 the **decision request**, etc., are not prohibited.

### 3262 7.2.3 Permit-biased PEP

3263 If the **decision** is "Deny", then the **PEP** SHALL deny **access**. If **obligations** accompany the **decision**,  
3264 then the **PEP** shall deny **access** only if it understands, and it can and will discharge those **obligations**.

3265 All other **decisions** SHALL result in the permission of **access**.

3266 Note: other actions, e.g. consultation of additional **PDPs**, reformulation/resubmission of  
3267 the **decision request**, etc., are not prohibited.

## 3268 7.3 Attribute evaluation

3269 **Attributes** are represented in the request **context** by the **context handler**, regardless of whether or not  
3270 they appeared in the original **decision request**, and are referred to in the **policy** by attribute designators  
3271 and attribute selectors. A **named attribute** is the term used for the criteria that the specific attribute  
3272 designators use to refer to particular **attributes** in the <Attributes> elements of the request **context**.

### 3273 7.3.1 Structured attributes

3274 <AttributeValue> elements MAY contain an instance of a structured XML data-type, for example  
3275 <ds:KeyInfo>. XACML 3.0 supports several ways for comparing the contents of such elements.

3276 1. In some cases, such elements MAY be compared using one of the XACML string functions, such  
3277 as "string-regexp-match", described below. This requires that the element be given the data-type  
3278 "http://www.w3.org/2001/XMLSchema#string". For example, a structured data-type that is  
3279 actually a ds:KeyInfo/KeyName would appear in the **Context** as:

```
3280 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">  
3281 &lt;ds:KeyName&gt;jhibbert-key&lt;/ds:KeyName&gt;  
3282 </AttributeValue>
```

3283 In general, this method will not be adequate unless the structured data-type is quite simple.

3284 2. The structured **attribute** MAY be made available in the <Content> element of the appropriate  
3285 **attribute** category and an <AttributeSelector> element MAY be used to select the contents  
3286 of a leaf sub-element of the structured data-type by means of an XPath expression. That value  
3287 MAY then be compared using one of the supported XACML functions appropriate for its primitive  
3288 data-type. This method requires support by the **PDP** for the optional XPath expressions feature.

3289 3. The structured **attribute** MAY be made available in the <Content> element of the appropriate  
3290 **attribute** category and an <AttributeSelector> element MAY be used to select any node in  
3291 the structured data-type by means of an XPath expression. This node MAY then be compared  
3292 using one of the XPath-based functions described in Section A.3.15. This method requires  
3293 support by the **PDP** for the optional XPath expressions and XPath functions features.

3294 See also Section 7.3.

### 3295 7.3.2 Attribute bags

3296 XACML defines implicit collections of its data-types. XACML refers to a collection of values that are of a  
3297 single data-type as a **bag**. **Bags** of data-types are needed because selections of nodes from an XML  
3298 **resource** or XACML request **context** may return more than one value.

3299 The <AttributeSelector> element uses an XPath expression to specify the selection of data from  
3300 free form XML. The result of an XPath expression is termed a node-set, which contains all the nodes  
3301 from the XML content that match the **predicate** in the XPath expression. Based on the various indexing  
3302 functions provided in the XPath specification, it SHALL be implied that a resultant node-set is the

3303 collection of the matching nodes. XACML also defines the <AttributeDesignator> element to have  
3304 the same matching methodology for **attributes** in the XACML request **context**.

3305 The values in a **bag** are not ordered, and some of the values may be duplicates. There SHALL be no  
3306 notion of a **bag** containing **bags**, or a **bag** containing values of differing types; i.e., a **bag** in XACML  
3307 SHALL contain only values that are of the same data-type.

### 3308 7.3.3 Multivalued attributes

3309 If a single <Attribute> element in a request **context** contains multiple <AttributeValue> child  
3310 elements, then the **bag** of values resulting from evaluation of the <Attribute> element MUST be  
3311 identical to the **bag** of values that results from evaluating a **context** in which each <AttributeValue>  
3312 element appears in a separate <Attribute> element, each carrying identical meta-data.

### 3313 7.3.4 Attribute Matching

3314 A **named attribute** includes specific criteria with which to match **attributes** in the **context**. An **attribute**  
3315 specifies a **Category**, **AttributeId** and **DataType**, and a **named attribute** also specifies the  
3316 **Issuer**. A **named attribute** SHALL match an **attribute** if the values of their respective **Category**,  
3317 **AttributeId**, **DataType** and optional **Issuer** attributes match. The **Category** of the **named**  
3318 **attribute** MUST match, by **identifier equality**, the **Category** of the corresponding **context attribute**.  
3319 The **AttributeId** of the **named attribute** MUST match, by **identifier equality**, the **AttributeId** of  
3320 the corresponding **context attribute**. The **DataType** of the **named attribute** MUST match, by **identifier**  
3321 **equality**, the **DataType** of the corresponding **context attribute**. If **Issuer** is supplied in the **named**  
3322 **attribute**, then it MUST match, using the urn:oasis:names:tc:xacml:1.0:function:string-equal function, the  
3323 **Issuer** of the corresponding **context attribute**. If **Issuer** is not supplied in the **named attribute**, then  
3324 the matching of the **context attribute** to the **named attribute** SHALL be governed by **AttributeId** and  
3325 **DataType** alone, regardless of the presence, absence, or actual value of **Issuer** in the corresponding  
3326 **context attribute**. In the case of an attribute selector, the matching of the **attribute** to the **named**  
3327 **attribute** SHALL be governed by the XPath expression and **DataType**.

### 3328 7.3.5 Attribute Retrieval

3329 The **PDP** SHALL request the values of **attributes** in the request **context** from the **context handler**. The  
3330 **context handler** MAY also add **attributes** to the request **context** without the **PDP** requesting them. The  
3331 **PDP** SHALL reference the **attributes** as if they were in a physical request **context** document, but the  
3332 **context handler** is responsible for obtaining and supplying the requested values by whatever means it  
3333 deems appropriate, including by retrieving them from one or more Policy Information Points. The **context**  
3334 **handler** SHALL return the values of **attributes** that match the attribute designator or attribute selector  
3335 and form them into a **bag** of values with the specified data-type. If no **attributes** from the request  
3336 **context** match, then the **attribute** SHALL be considered missing. If the **attribute** is missing, then  
3337 **MustBePresent** governs whether the attribute designator or attribute selector returns an empty **bag** or  
3338 an "Indeterminate" result. If **MustBePresent** is "False" (default value), then a missing **attribute** SHALL  
3339 result in an empty **bag**. If **MustBePresent** is "True", then a missing **attribute** SHALL result in  
3340 "Indeterminate". This "Indeterminate" result SHALL be handled in accordance with the specification of the  
3341 encompassing expressions, **rules**, **policies** and **policy sets**. If the result is "Indeterminate", then the  
3342 **AttributeId**, **DataType** and **Issuer** of the **attribute** MAY be listed in the **authorization decision** as  
3343 described in Section 7.17. However, a **PDP** MAY choose not to return such information for security  
3344 reasons.

3345 Regardless of any dynamic modifications of the request **context** during policy evaluation, the **PDP**  
3346 SHALL behave as if each bag of **attribute** values is fully populated in the **context** before it is first tested,  
3347 and is thereafter immutable during evaluation. (That is, every subsequent test of that **attribute** shall use  
3348 the same bag of values that was initially tested.)

### 3349 7.3.6 Environment Attributes

3350 Standard **environment attributes** are listed in Section B.7. If a value for one of these **attributes** is  
3351 supplied in the **decision request**, then the **context handler** SHALL use that value. Otherwise, the  
3352 **context handler** SHALL supply a value. In the case of date and time **attributes**, the supplied value  
3353 SHALL have the semantics of the "date and time that apply to the **decision request**".

### 3354 7.3.7 AttributeSelector evaluation

3355 An <AttributeSelector> element will be evaluated according to the following processing model.

3356

3357 NOTE: It is not necessary for an implementation to actually follow these steps. It is only  
3358 necessary to produce results identical to those that would be produced by following these  
3359 steps.

3360 1. If the **attributes** category given by the **Category** attribute is not found or does not have a  
3361 <Content> child element, then the return value is either "Indeterminate" or an empty **bag** as  
3362 determined by the **MustBePresent** attribute; otherwise, construct an XML data structure  
3363 suitable for xpath processing from the <Content> element in the attributes category given by the  
3364 Category attribute. The data structure shall be constructed so that the document node of this  
3365 structure contains a single document element which corresponds to the single child element of  
3366 the <Content> element. The constructed data structure shall be equivalent to one that would  
3367 result from parsing a stand-alone XML document consisting of the contents of the <Content>  
3368 element (including any comment and processing-instruction markup). Namespace declarations  
3369 which are not "visibly utilized", as defined by [exc-c14n], MAY not be present and MUST NOT be  
3370 utilized by the XPath expression in step 3. The data structure must meet the requirements of the  
3371 applicable xpath version.

3372 2. Select a context node for xpath processing from this data structure. If there is a  
3373 **ContextSelectorId** attribute, the context node shall be the node selected by applying the  
3374 XPath expression given in the **attribute** value of the designated **attribute** (in the **attributes**  
3375 category given by the <AttributeSelector> **Category** attribute). It shall be an error if this  
3376 evaluation returns no node or more than one node, in which case the return value MUST be an  
3377 "Indeterminate" with a status code "urn:oasis:names:tc:xacml:1.0:status:syntax-error". If there is  
3378 no **ContextSelectorId**, the document node of the data structure shall be the context node.

3379 3. Evaluate the XPath expression given in the **Path** attribute against the xml data structure, using  
3380 the context node selected in the previous step. It shall be an error if this evaluation returns  
3381 anything other than a sequence of nodes (possibly empty), in which case the  
3382 <AttributeSelector> MUST return "Indeterminate" with a status code  
3383 "urn:oasis:names:tc:xacml:1.0:status:syntax-error". If the evaluation returns an empty sequence  
3384 of nodes, then the return value is either "Indeterminate" or an empty **bag** as determined by the  
3385 **MustBePresent** attribute.

3386 4. If the data type is a primitive data type, convert the text value of each selected node to the  
3387 desired data type, as specified in the **DataType** attribute. Each value shall be constructed using  
3388 the appropriate constructor function from [XF] Section 5 listed below, corresponding to the  
3389 specified data type.

3390  
3391 xs:string()  
3392 xs:boolean()  
3393 xs:integer()  
3394 xs:double()  
3395 xs:dateTime()  
3396 xs:date()  
3397 xs:time()  
3398 xs:hexBinary()  
3399 xs:base64Binary()

Deleted: Construct

Formatted: Default Paragraph Font

Deleted:

Formatted: Attribute

3402 xs:anyURI()  
3403 xs:yearMonthDuration()  
3404 xs:dayTimeDuration()  
3405

3406 If the `DataType` is not one of the primitive types listed above, then the return values shall be  
3407 constructed from the nodeset in a manner specified by the particular `DataType` extension  
3408 specification. If the data type extension does not specify an appropriate constructor function, then  
3409 the `<AttributeSelector>` MUST return "Indeterminate" with a status code  
3410 "urn:oasis:names:tc:xacml:1.0:status:syntax-error".  
3411

3412 If an error occurs when converting the values returned by the XPath expression to the specified  
3413 `DataType`, then the result of the `<AttributeSelector>` MUST be "Indeterminate", with a  
3414 status code "urn:oasis:names:tc:xacml:1.0:status:processing-error"

## 3415 7.4 Expression evaluation

3416 XACML specifies expressions in terms of the elements listed below, of which the `<Apply>` and  
3417 `<Condition>` elements recursively compose greater expressions. Valid expressions SHALL be type  
3418 correct, which means that the types of each of the elements contained within `<Apply>` elements SHALL  
3419 agree with the respective argument types of the function that is named by the `FunctionId` attribute.  
3420 The resultant type of the `<Apply>` element SHALL be the resultant type of the function, which MAY be  
3421 narrowed to a primitive data-type, or a **bag** of a primitive data-type, by type-unification. XACML defines  
3422 an evaluation result of "Indeterminate", which is said to be the result of an invalid expression, or an  
3423 operational error occurring during the evaluation of the expression.

3424 XACML defines these elements to be in the substitution group of the `<Expression>` element:

- 3425 • `<xacml:AttributeValue>`
- 3426 • `<xacml:AttributeDesignator>`
- 3427 • `<xacml:AttributeSelector>`
- 3428 • `<xacml:Apply>`
- 3429 • `<xacml:Function>`
- 3430 • `<xacml:VariableReference>`

## 3431 7.5 Arithmetic evaluation

3432 IEEE 754 [IEEE754] specifies how to evaluate arithmetic functions in a context, which specifies defaults  
3433 for precision, rounding, etc. XACML SHALL use this specification for the evaluation of all integer and  
3434 double functions relying on the Extended Default Context, enhanced with double precision:

3435 flags - all set to 0

3436 trap-enablers - all set to 0 (IEEE 854 §7) with the exception of the "division-by-zero" trap enabler,  
3437 which SHALL be set to 1

3438 precision - is set to the designated double precision

3439 rounding - is set to round-half-even (IEEE 854 §4.1)

## 3440 7.6 Match evaluation

3441 The **attribute** matching element `<Match>` appears in the `<Target>` element of **rules**, **policies** and  
3442 **policy sets**.

3443 This element represents a Boolean expression over **attributes** of the request **context**. A matching  
3444 element contains a `MatchId` attribute that specifies the function to be used in performing the match  
3445 evaluation, an `<AttributeValue>` and an `<AttributeDesignator>` or `<AttributeSelector>`  
3446 element that specifies the **attribute** in the **context** that is to be matched against the specified value.



3447 The `MatchId` attribute SHALL specify a function that takes two arguments, returning a result type of  
3448 "http://www.w3.org/2001/XMLSchema#boolean". The **attribute** value specified in the matching element  
3449 SHALL be supplied to the `MatchId` function as its first argument. An element of the **bag** returned by the  
3450 `<AttributeDesignator>` or `<AttributeSelector>` element SHALL be supplied to the `MatchId`  
3451 function as its second argument, as explained below. The `DataType` of the `<AttributeValue>`  
3452 SHALL match the data-type of the first argument expected by the `MatchId` function. The `DataType` of  
3453 the `<AttributeDesignator>` or `<AttributeSelector>` element SHALL match the data-type of the  
3454 second argument expected by the `MatchId` function.

3455 In addition, functions that are strictly within an extension to XACML MAY appear as a value for the  
3456 `MatchId` attribute, and those functions MAY use data-types that are also extensions, so long as the  
3457 extension function returns a Boolean result and takes two single base types as its inputs. The function  
3458 used as the value for the `MatchId` attribute SHOULD be easily indexable. Use of non-indexable or  
3459 complex functions may prevent efficient evaluation of **decision requests**.

3460 The evaluation semantics for a matching element is as follows. If an operational error were to occur while  
3461 evaluating the `<AttributeDesignator>` or `<AttributeSelector>` element, then the result of the  
3462 entire expression SHALL be "Indeterminate". If the `<AttributeDesignator>` or  
3463 `<AttributeSelector>` element were to evaluate to an empty **bag**, then the result of the expression  
3464 SHALL be "False". Otherwise, the `MatchId` function SHALL be applied between the  
3465 `<AttributeValue>` and each element of the **bag** returned from the `<AttributeDesignator>` or  
3466 `<AttributeSelector>` element. If at least one of those function applications were to evaluate to  
3467 "True", then the result of the entire expression SHALL be "True". Otherwise, if at least one of the function  
3468 applications results in "Indeterminate", then the result SHALL be "Indeterminate". Finally, if all function  
3469 applications evaluate to "False", then the result of the entire expression SHALL be "False".

3470 It is also possible to express the semantics of a **target** matching element in a **condition**. For instance,  
3471 the **target** match expression that compares a "**subject-name**" starting with the name "John" can be  
3472 expressed as follows:

```
3473 <Match  
3474 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">  
3475   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">  
3476     John.*  
3477   </AttributeValue>  
3478   <AttributeDesignator  
3479     Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-  
3480 subject"  
3481     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"  
3482     DataType="http://www.w3.org/2001/XMLSchema#string"/>  
3483 </Match>
```

3484 Alternatively, the same match semantics can be expressed as an `<Apply>` element in a **condition** by  
3485 using the "urn:oasis:names:tc:xacml:3.0:function:any-of" function, as follows:

```
3486 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">  
3487   <Function  
3488     FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match"/>  
3489   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">  
3490     John.*  
3491   </AttributeValue>  
3492   <AttributeDesignator  
3493     Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-  
3494 subject"  
3495     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"  
3496     DataType="http://www.w3.org/2001/XMLSchema#string"/>  
3497 </Apply>
```

3498 **7.7 Target evaluation**

3499 An empty **target** matches any request. Otherwise the **target** value SHALL be "Match" if all the AnyOf  
 3500 specified in the **target** match values in the request **context**. Otherwise, if any one of the AnyOf specified  
 3501 in the **target** is "No Match", then the **target** SHALL be "No Match". Otherwise, the **target** SHALL be  
 3502 "Indeterminate". The **target** match table is shown in Table 1.

<AnyOf> values	Target value
All "Match"	"Match"
At least one "No Match"	"No Match"
Otherwise	"Indeterminate"

3503 Table 1 Target match table

3504 The AnyOf SHALL match values in the request **context** if at least one of their <AllOf> elements  
 3505 matches a value in the request **context**. The AnyOf table is shown in Table 2.

<AllOf> values	<AnyOf> Value
At least one "Match"	"Match"
None matches and at least one "Indeterminate"	"Indeterminate"
All "No match"	"No match"

3506 Table 2 AnyOf match table

3507 An AllOf SHALL match a value in the request **context** if the value of all its <Match> elements is "True".  
 3508 The AllOf table is shown in Table 3.

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

3509 Table 3 AllOf match table

3510 **7.8 VariableReference Evaluation**

3511 The <VariableReference> element references a single <VariableDefinition> element contained  
 3512 within the same <Policy> element. A <VariableReference> that does not reference a particular  
 3513 <VariableDefinition> element within the encompassing <Policy> element is called an undefined  
 3514 reference. **Policies** with undefined references are invalid.

3515 In any place where a <VariableReference> occurs, it has the effect as if the text of the  
 3516 <Expression> element defined in the <VariableDefinition> element replaces the  
 3517 <VariableReference> element. Any evaluation scheme that preserves this semantic is acceptable.  
 3518 For instance, the expression in the <VariableDefinition> element may be evaluated to a particular  
 3519 value and cached for multiple references without consequence. (I.e. the value of an <Expression>  
 3520 element remains the same for the entire **policy** evaluation.) This characteristic is one of the benefits of  
 3521 XACML being a declarative language.

3522 A variable reference containing circular references is invalid. The PDP MUST detect circular references  
 3523 either at policy loading time or during runtime evaluation. If the PDP detects a circular reference during

3524 runtime the variable reference evaluates to "Indeterminate" with status code  
 3525 urn:oasis:names:tc:xacml:1.0:status:processing-error.

## 3526 7.9 Condition evaluation

3527 The **condition** value SHALL be "True" if the <Condition> element is absent, or if it evaluates to "True".  
 3528 Its value SHALL be "False" if the <Condition> element evaluates to "False". The **condition** value  
 3529 SHALL be "Indeterminate", if the expression contained in the <Condition> element evaluates to  
 3530 "Indeterminate."

## 3531 7.10 Extended Indeterminate

3532 Some **combining algorithms** are defined in terms of an extended set of "Indeterminate" values. The  
 3533 extended set associated with the "Indeterminate" contains the potential effect values which could have  
 3534 occurred if there would not have been an error causing the "Indeterminate". The possible extended set  
 3535 "Indeterminate" values are

- 3536 • "Indeterminate{D}": an "Indeterminate" from a **policy** or **rule** which could have evaluated to "Deny",  
 3537 but not "Permit"
- 3538 • "Indeterminate{P}": an "Indeterminate" from a **policy** or **rule** which could have evaluated to "Permit",  
 3539 but not "Deny"
- 3540 • "Indeterminate{DP}": an "Indeterminate" from a **policy** or **rule** which could have evaluated to "Deny"  
 3541 or "Permit".

3542 The **combining algorithms** which are defined in terms of the extended "Indeterminate" make use of the  
 3543 additional information to allow for better treatment of errors in the algorithms.

3544 The final decision returned by a **PDP** cannot be an extended Indeterminate. Any such decision at the top  
 3545 level **policy** or **policy set** is returned as a plain Indeterminate in the response from the **PDP**.

3546 The tables in the following four sections define how extended "Indeterminate" values are produced during  
 3547 **Rule**, **Policy** and **PolicySet** evaluation.

## 3548 7.11 Rule evaluation

3549 A **rule** has a value that can be calculated by evaluating its contents. **Rule** evaluation involves separate  
 3550 evaluation of the **rule's target** and **condition**. The **rule** truth table is shown in Table 4.

Target	Condition	Rule Value
"Match" or no target	"True"	<b>Effect</b>
"Match" or no target	"False"	"NotApplicable"
"Match" or no target	"Indeterminate"	"Indeterminate{P}" if the <b>Effect</b> is Permit, or "Indeterminate{D}" if the <b>Effect</b> is Deny
"No-match"	Don't care	"NotApplicable"
"Indeterminate"	Don't care	"Indeterminate{P}" if the <b>Effect</b> is Permit, or "Indeterminate{D}" if the <b>Effect</b> is Deny

3551 Table 4 Rule truth table.

## 3552 7.12 Policy evaluation

3553 The value of a **policy** SHALL be determined only by its contents, considered in relation to the contents of  
 3554 the request **context**. A **policy's** value SHALL be determined by evaluation of the **policy's target** and,  
 3555 according to the specified **rule-combining algorithm, rules**.

3556 The **policy** truth table is shown in Table 5.

<b>Target</b>	<b>Rule values</b>	<b>Policy Value</b>
"Match"	Don't care	Specified by the <b>rule-combining algorithm</b>
"No-match"	Don't care	"NotApplicable"
"Indeterminate"	See Table 7	See Table 7

3557 Table 5 Policy truth table

3558 Note that none of the **rule-combining algorithms** defined by XACML 3.0 take parameters. However,  
 3559 non-standard combining algorithms MAY take parameters. In such a case, the values of these  
 3560 parameters associated with the **rules**, MUST be taken into account when evaluating the **policy**. The  
 3561 parameters and their types should be defined in the specification of the combining algorithm. If the  
 3562 implementation supports combiner parameters and if combiner parameters are present in a **policy**, then  
 3563 the parameter values MUST be supplied to the combining algorithm implementation.

### 3564 7.13 Policy Set evaluation

3565 The value of a **policy set** SHALL be determined by its contents, considered in relation to the contents of  
 3566 the request **context**. A **policy set's** value SHALL be determined by evaluation of the **policy set's target**,  
 3567 and, according to the specified **policy-combining algorithm, policies** and **policy sets**,  
 3568 The **policy set** truth table is shown in Table 6.

<b>Target</b>	<b>Policy values</b>	<b>Policy set Value</b>
"Match"	Don't care	Specified by the <b>policy-combining algorithm</b>
"No-match"	Don't care	"NotApplicable"
"Indeterminate"	See Table 7	See Table 7

3569 Table 6 Policy set truth table

3570 Note that none of the **policy-combining algorithms** defined by XACML 3.0 take parameters. However,  
 3571 non-standard combining algorithms MAY take parameters. In such a case, the values of these  
 3572 parameters associated with the **policies**, MUST be taken into account when evaluating the **policy set**.  
 3573 The parameters and their types should be defined in the specification of the combining algorithm. If the  
 3574 implementation supports combiner parameters and if combiner parameters are present in a **policy**, then  
 3575 the parameter values MUST be supplied to the combining algorithm implementation.

### 3576 7.14 Policy and Policy set value for Indeterminate Target

3577 If the **target** of a **policy** or **policy set** evaluates to "Indeterminate", the value of the **policy** or **policy set**  
 3578 as a whole is determined by the value of the **combining algorithm** according to Table 7.

<b>Combining algorithm Value</b>	<b>Policy set or policy Value</b>
"NotApplicable"	"NotApplicable"
"Permit"	"Indeterminate{P}"
"Deny"	"Indeterminate{D}"
"Indeterminate"	"Indeterminate{DP}"
"Indeterminate{DP}"	"Indeterminate{DP}"
"Indeterminate{P}"	"Indeterminate{P}"

"Indeterminate{D}"	"Indeterminate{D}"
--------------------	--------------------

3579 Table 7 The value of a policy or policy set when the target is "Indeterminate".

## 3580 7.15 PolicySetIdReference and PolicyIdReference evaluation

3581 A policy set id reference or a policy id reference is evaluated by resolving the reference and evaluating  
3582 the referenced policy set or policy.

3583 If resolving the reference fails, the reference evaluates to "Indeterminate" with status code  
3584 urn:oasis:names:tc:xacml:1.0:status:processing-error.

3585 A policy set id reference or a policy id reference containing circular references is invalid. The PDP MUST  
3586 detect circular references either at policy loading time or during runtime evaluation. If the PDP detects a  
3587 circular reference during runtime the reference evaluates to "Indeterminate" with status code  
3588 urn:oasis:names:tc:xacml:1.0:status:processing-error.

## 3589 7.16 Hierarchical resources

3590 It is often the case that a **resource** is organized as a hierarchy (e.g. file system, XML document). XACML  
3591 provides several optional mechanisms for supporting hierarchical **resources**. These are described in the  
3592 XACML Profile for Hierarchical Resources [**Hier**] and in the XACML Profile for Requests for Multiple  
3593 Resources [**Multi**].

## 3594 7.17 Authorization decision

3595 In relation to a particular **decision request**, the **PDP** is defined by a **policy-combining algorithm** and a  
3596 set of **policies** and/or **policy sets**. The **PDP** SHALL return a response **context** as if it had evaluated a  
3597 single **policy set** consisting of this **policy-combining algorithm** and the set of **policies** and/or **policy**  
3598 **sets**.

3599 The **PDP** MUST evaluate the **policy set** as specified in Sections 5 and 7. The **PDP** MUST return a  
3600 response **context**, with one <Decision> element of value "Permit", "Deny", "Indeterminate" or  
3601 "NotApplicable".

3602 If the **PDP** cannot make a **decision**, then an "Indeterminate" <Decision> element SHALL be returned.

## 3603 7.18 Obligations and advice

3604 A **rule**, **policy**, or **policy set** may contain one or more **obligation** or **advice** expressions. When such a  
3605 **rule**, **policy**, or **policy set** is evaluated, the **obligation** or **advice** expression SHALL be evaluated to an  
3606 **obligation** or **advice** respectively, which SHALL be passed up to the next level of evaluation (the  
3607 enclosing or referencing **policy**, **policy set**, or **authorization decision**) only if the result of the **rule**,  
3608 **policy**, or **policy set** being evaluated matches the value of the FulfillOn attribute of the **obligation** or  
3609 the AppliesTo attribute of the **advice**. If any of the **attribute** assignment expressions in an **obligation**  
3610 or **advice** expression with a matching FulfillOn or AppliesTo attribute evaluates to "Indeterminate",  
3611 then the whole **rule**, **policy**, or **policy set** SHALL be "Indeterminate". If the FulfillOn or AppliesTo  
3612 attribute does not match the result of the combining algorithm or the **rule** evaluation, then any  
3613 indeterminate in an **obligation** or **advice** expression has no effect.

3614 As a consequence of this procedure, no **obligations** or **advice** SHALL be returned to the **PEP** if the **rule**,  
3615 **policies**, or **policy sets** from which they are drawn are not evaluated, or if their evaluated result is  
3616 "Indeterminate" or "NotApplicable", or if the **decision** resulting from evaluating the **rule**, **policy**, or **policy**  
3617 **set** does not match the **decision** resulting from evaluating an enclosing **policy set**.

3618 If the **PDP**'s evaluation is viewed as a tree of **rules**, **policy sets** and **policies**, each of which returns  
3619 "Permit" or "Deny", then the set of **obligations** and **advice** returned by the **PDP** to the **PEP** will include  
3620 only the **obligations** and **advice** associated with those paths where the result at each level of evaluation  
3621 is the same as the result being returned by the **PDP**. In situations where any lack of determinism is  
3622 unacceptable, a deterministic combining algorithm, such as ordered-deny-overrides, should be used.

3623 Also see Section 7.2.

## 3624 7.19 Exception handling

3625 XACML specifies behavior for the **PDP** in the following situations.

### 3626 7.19.1 Unsupported functionality

3627 If the **PDP** attempts to evaluate a **policy set** or **policy** that contains an optional element type or function  
3628 that the **PDP** does not support, then the **PDP** SHALL return a <Decision> value of "Indeterminate". If a  
3629 <StatusCode> element is also returned, then its value SHALL be  
3630 "urn:oasis:names:tc:xacml:1.0:status:syntax-error" in the case of an unsupported element type, and  
3631 "urn:oasis:names:tc:xacml:1.0:status:processing-error" in the case of an unsupported function.

### 3632 7.19.2 Syntax and type errors

3633 If a **policy** that contains invalid syntax is evaluated by the XACML **PDP** at the time a **decision request** is  
3634 received, then the result of that **policy** SHALL be "Indeterminate" with a **StatusCode** value of  
3635 "urn:oasis:names:tc:xacml:1.0:status:syntax-error".  
3636 If a **policy** that contains invalid static data-types is evaluated by the XACML **PDP** at the time a **decision**  
3637 **request** is received, then the result of that **policy** SHALL be "Indeterminate" with a **StatusCode** value of  
3638 "urn:oasis:names:tc:xacml:1.0:status:processing-error".

### 3639 7.19.3 Missing attributes

3640 The absence of matching **attributes** in the request **context** for any of the attribute designators attribute or  
3641 selectors that are found in the **policy** will result in an enclosing <AllOf> element to return a value of  
3642 "Indeterminate", if the designator or selector has the **MustBePresent** XML attribute set to true, as  
3643 described in Sections 5.29 and 5.30 and may result in a <Decision> element containing the  
3644 "Indeterminate" value. If, in this case a status code is supplied, then the value  
3645 "urn:oasis:names:tc:xacml:1.0:status:missing-attribute"  
3646 SHALL be used, to indicate that more information is needed in order for a definitive **decision** to be  
3647 rendered. In this case, the <Status> element MAY list the names and data-types of any **attributes** that  
3648 are needed by the **PDP** to refine its **decision** (see Section 5.58). A **PEP** MAY resubmit a refined request  
3649 **context** in response to a <Decision> element contents of "Indeterminate" with a status code of  
3650 "urn:oasis:names:tc:xacml:1.0:status:missing-attribute"  
3651 by adding **attribute** values for the **attribute** names that were listed in the previous response. When the  
3652 **PDP** returns a <Decision> element contents of "Indeterminate", with a status code of  
3653 "urn:oasis:names:tc:xacml:1.0:status:missing-attribute",  
3654 it MUST NOT list the names and data-types of any **attribute** for which values were supplied in the original  
3655 request. Note, this requirement forces the **PDP** to eventually return an **authorization decision** of  
3656 "Permit", "Deny", or "Indeterminate" with some other status code, in response to successively-refined  
3657 requests.

## 3658 7.20 Identifier equality

3659 XACML makes use of URIs and strings as identifiers. When such identifiers are compared for equality,  
3660 the comparison MUST be done so that the identifiers are equal if they have the same length and the  
3661 characters in the two identifiers are equal codepoint by codepoint.  
3662 The following is a list of the identifiers which MUST use this definition of equality.  
3663 The content of the element <XPathVersion>.  
3664 The XML attribute **Value** in the element <StatusCode>.

- 3665 The XML attributes `Category`, `AttributeId`, `DataType` and `Issuer` in the element  
3666 `<MissingAttributeDetail>`.
- 3667 The XML attribute `Category` in the element `<Attributes>`.
- 3668 The XML attributes `AttributeId` and `Issuer` in the element `<Attribute>`.
- 3669 The XML attribute `ObligationId` in the element `<Obligation>`.
- 3670 The XML attribute `AdviceId` in the element `<Advice>`.
- 3671 The XML attributes `AttributeId` and `Category` in the element `<AttributeAssignment>`.
- 3672 The XML attribute `ObligationId` in the element `<ObligationExpression>`.
- 3673 The XML attribute `AdviceId` in the element `<AdviceExpression>`.
- 3674 The XML attributes `AttributeId`, `Category` and `Issuer` in the element  
3675 `<AttributeAssignmentExpression>`.
- 3676 The XML attributes `PolicySetId` and `PolicyCombiningAlgId` in the element `<PolicySet>`.
- 3677 The XML attribute `ParameterName` in the element `<CombinerParameter>`.
- 3678 The XML attribute `RuleIdRef` in the element `<RuleCombinerParameters>`.
- 3679 The XML attribute `PolicyIdRef` in the element `<PolicyCombinerParameters>`.
- 3680 The XML attribute `PolicySetIdRef` in the element `<PolicySetCombinerParameters>`.
- 3681 The anyURI in the content of the complex type `IdReferenceType`.
- 3682 The XML attributes `PolicyId` and `RuleCombiningAlgId` in the element `<Policy>`.
- 3683 The XML attribute `RuleId` in the element `<Rule>`.
- 3684 The XML attribute `MatchId` in the element `<Match>`.
- 3685 The XML attribute `VariableId` in the element `<VariableDefinition>`.
- 3686 The XML attribute `VariableId` in the element `<VariableReference>`.
- 3687 The XML attributes `Category`, `ContextSelectorId` and `DataType` in the element  
3688 `<AttributeSelector>`.
- 3689 The XML attributes `Category`, `AttributeId`, `DataType` and `Issuer` in the element  
3690 `<AttributeDesignator>`.
- 3691 The XML attribute `DataType` in the element `<AttributeValue>`.
- 3692 The XML attribute `FunctionId` in the element `<Function>`.
- 3693 The XML attribute `FunctionId` in the element `<Apply>`.
- 3694
- 3695 It is RECOMMENDED that extensions to XACML use the same definition of identifier equality for similar  
3696 identifiers.
- 3697 It is RECOMMENDED that extensions which define identifiers do not define identifiers which could be  
3698 easily misinterpreted by people as being subject to other kind of processing, such as URL character  
3699 escaping, before matching.

---

## 3700 8 XACML extensibility points (non-normative)

3701 This section describes the points within the XACML model and schema where extensions can be added.

### 3702 8.1 Extensible XML attribute types

3703 The following XML attributes have values that are URIs. These may be extended by the creation of new  
3704 URIs associated with new semantics for these attributes.

3705 Category,

3706 AttributeId,

3707 DataType,

3708 FunctionId,

3709 MatchId,

3710 ObligationId,

3711 AdviceId,

3712 PolicyCombiningAlgId,

3713 RuleCombiningAlgId,

3714 StatusCode,

3715 SubjectCategory.

3716 See Section 5 for definitions of these *attribute* types.

### 3717 8.2 Structured attributes

3718 <AttributeValue> elements MAY contain an instance of a structured XML data-type. Section 7.3.1  
3719 describes a number of standard techniques to identify data items within such a structured *attribute*.  
3720 Listed here are some additional techniques that require XACML extensions.

- 3721 1. For a given structured data-type, a community of XACML users MAY define new *attribute*  
3722 identifiers for each leaf sub-element of the structured data-type that has a type conformant with  
3723 one of the XACML-defined primitive data-types. Using these new *attribute* identifiers, the *PEPs*  
3724 or *context handlers* used by that community of users can flatten instances of the structured  
3725 data-type into a sequence of individual <Attribute> elements. Each such <Attribute>  
3726 element can be compared using the XACML-defined functions. Using this method, the structured  
3727 data-type itself never appears in an <AttributeValue> element.
- 3728 2. A community of XACML users MAY define a new function that can be used to compare a value of  
3729 the structured data-type against some other value. This method may only be used by *PDPs* that  
3730 support the new function.



---

3731 **9 Security and privacy considerations (non-**  
3732 **normative)**

3733 This section identifies possible security and privacy compromise scenarios that should be considered  
3734 when implementing an XACML-based system. The section is informative only. It is left to the  
3735 implementer to decide whether these compromise scenarios are practical in their environment and to  
3736 select appropriate safeguards.

3737 **9.1 Threat model**

3738 We assume here that the adversary has access to the communication channel between the XACML  
3739 actors and is able to interpret, insert, delete, and modify messages or parts of messages.

3740 Additionally, an actor may use information from a former message maliciously in subsequent transactions.  
3741 It is further assumed that **rules** and **policies** are only as reliable as the actors that create and use them.  
3742 Thus it is incumbent on each actor to establish appropriate trust in the other actors upon which it relies.  
3743 Mechanisms for trust establishment are outside the scope of this specification.

3744 The messages that are transmitted between the actors in the XACML model are susceptible to attack by  
3745 malicious third parties. Other points of vulnerability include the **PEP**, the **PDP**, and the **PAP**. While some  
3746 of these entities are not strictly within the scope of this specification, their compromise could lead to the  
3747 compromise of **access control** enforced by the **PEP**.

3748 It should be noted that there are other components of a distributed system that may be compromised,  
3749 such as an operating system and the domain-name system (DNS) that are outside the scope of this  
3750 discussion of threat models. Compromise in these components may also lead to a policy violation.

3751 The following sections detail specific compromise scenarios that may be relevant to an XACML system.

3752 **9.1.1 Unauthorized disclosure**

3753 XACML does not specify any inherent mechanisms to protect the confidentiality of the messages  
3754 exchanged between actors. Therefore, an adversary could observe the messages in transit. Under  
3755 certain security **policies**, disclosure of this information is a violation. Disclosure of **attributes** or the types  
3756 of **decision requests** that a **subject** submits may be a breach of privacy policy. In the commercial  
3757 sector, the consequences of unauthorized disclosure of personal data may range from embarrassment to  
3758 the custodian, to imprisonment and/or large fines in the case of medical or financial data.

3759 Unauthorized disclosure is addressed by confidentiality safeguards.

3760 **9.1.2 Message replay**

3761 A message replay attack is one in which the adversary records and replays legitimate messages between  
3762 XACML actors. This attack may lead to denial of service, the use of out-of-date information or  
3763 impersonation.

3764 Prevention of replay attacks requires the use of message freshness safeguards.

3765 Note that encryption of the message does not mitigate a replay attack since the message is simply  
3766 replayed and does not have to be understood by the adversary.

3767 **9.1.3 Message insertion**

3768 A message insertion attack is one in which the adversary inserts messages in the sequence of messages  
3769 between XACML actors.

3770 The solution to a message insertion attack is to use mutual authentication and message sequence  
3771 integrity safeguards between the actors. It should be noted that just using SSL mutual authentication is  
3772 not sufficient. This only proves that the other party is the one identified by the **subject** of the X.509

3773 certificate. In order to be effective, it is necessary to confirm that the certificate **subject** is authorized to  
3774 send the message.

#### 3775 9.1.4 Message deletion

3776 A message deletion attack is one in which the adversary deletes messages in the sequence of messages  
3777 between XACML actors. Message deletion may lead to denial of service. However, a properly designed  
3778 XACML system should not render an incorrect **authorization decision** as a result of a message deletion  
3779 attack.

3780 The solution to a message deletion attack is to use message sequence integrity safeguards between the  
3781 actors.

#### 3782 9.1.5 Message modification

3783 If an adversary can intercept a message and change its contents, then they may be able to alter an  
3784 **authorization decision**. A message integrity safeguard can prevent a successful message modification  
3785 attack.

#### 3786 9.1.6 NotApplicable results

3787 A result of "NotApplicable" means that the **PDP** could not locate a **policy** whose **target** matched the  
3788 information in the **decision request**. In general, it is highly recommended that a "Deny" **effect policy** be  
3789 used, so that when a **PDP** would have returned "NotApplicable", a result of "Deny" is returned instead.

3790 In some security models, however, such as those found in many web servers, an **authorization decision**  
3791 of "NotApplicable" is treated as equivalent to "Permit". There are particular security considerations that  
3792 must be taken into account for this to be safe. These are explained in the following paragraphs.

3793 If "NotApplicable" is to be treated as "Permit", it is vital that the matching algorithms used by the **policy** to  
3794 match elements in the **decision request** be closely aligned with the data syntax used by the applications  
3795 that will be submitting the **decision request**. A failure to match will result in "NotApplicable" and be  
3796 treated as "Permit". So an unintended failure to match may allow unintended **access**.

3797 Commercial http responders allow a variety of syntaxes to be treated equivalently. The "%" can be used  
3798 to represent characters by hex value. The URL path "/../" provides multiple ways of specifying the same  
3799 value. Multiple character sets may be permitted and, in some cases, the same printed character can be  
3800 represented by different binary values. Unless the matching algorithm used by the **policy** is sophisticated  
3801 enough to catch these variations, unintended **access** may be permitted.

3802 It may be safe to treat "NotApplicable" as "Permit" only in a closed environment where all applications that  
3803 formulate a **decision request** can be guaranteed to use the exact syntax expected by the **policies**. In a  
3804 more open environment, where **decision requests** may be received from applications that use any legal  
3805 syntax, it is strongly recommended that "NotApplicable" NOT be treated as "Permit" unless matching  
3806 **rules** have been very carefully designed to match all possible applicable inputs, regardless of syntax or  
3807 type variations. Note, however, that according to Section 7.2, a **PEP** must deny **access** unless it  
3808 receives an explicit "Permit" **authorization decision**.

#### 3809 9.1.7 Negative rules

3810 A negative **rule** is one that is based on a **predicate** not being "True". If not used with care, negative  
3811 **rules** can lead to policy violations, therefore some authorities recommend that they not be used.  
3812 However, negative **rules** can be extremely efficient in certain cases, so XACML has chosen to include  
3813 them. Nevertheless, it is recommended that they be used with care and avoided if possible.

3814 A common use for negative **rules** is to deny **access** to an individual or subgroup when their membership  
3815 in a larger group would otherwise permit them **access**. For example, we might want to write a **rule** that  
3816 allows all vice presidents to see the unpublished financial data, except for Joe, who is only a ceremonial  
3817 vice president and can be indiscreet in his communications. If we have complete control over the  
3818 administration of **subject attributes**, a superior approach would be to define "Vice President" and  
3819 "Ceremonial Vice President" as distinct groups and then define **rules** accordingly. However, in some

3820 environments this approach may not be feasible. (It is worth noting in passing that referring to individuals  
3821 in **rules** does not scale well. Generally, shared **attributes** are preferred.)

3822 If not used with care, negative **rules** can lead to policy violations in two common cases: when **attributes**  
3823 are suppressed and when the base group changes. An example of suppressed **attributes** would be if we  
3824 have a **policy** that **access** should be permitted, unless the **subject** is a credit risk. If it is possible that  
3825 the **attribute** of being a credit risk may be unknown to the **PDP** for some reason, then unauthorized  
3826 **access** may result. In some environments, the **subject** may be able to suppress the publication of  
3827 **attributes** by the application of privacy controls, or the server or repository that contains the information  
3828 may be unavailable for accidental or intentional reasons.

3829 An example of a changing base group would be if there is a **policy** that everyone in the engineering  
3830 department may change software source code, except for secretaries. Suppose now that the department  
3831 was to merge with another engineering department and the intent is to maintain the same **policy**.  
3832 However, the new department also includes individuals identified as administrative assistants, who ought  
3833 to be treated in the same way as secretaries. Unless the **policy** is altered, they will unintentionally be  
3834 permitted to change software source code. Problems of this type are easy to avoid when one individual  
3835 administers all **policies**, but when administration is distributed, as XACML allows, this type of situation  
3836 must be explicitly guarded against.

### 3837 9.1.8 Denial of service

3838 A denial of service attack is one in which the adversary overloads an XACML actor with excessive  
3839 computations or network traffic such that legitimate users cannot access the services provided by the  
3840 actor.

3841 The urn:oasis:names:tc:xacml:3.0:function:access-permitted function may lead to hard to predict behavior  
3842 in the **PDP**. It is possible that the function is invoked during the recursive invocations of the **PDP** such that  
3843 loops are formed. Such loops may in some cases lead to large numbers of requests to be generated  
3844 before the **PDP** can detect the loop and abort evaluation. Such loops could cause a denial of service at  
3845 the **PDP**, either because of a malicious **policy** or because of a mistake in a **policy**.

## 3846 9.2 Safeguards

### 3847 9.2.1 Authentication

3848 Authentication provides the means for one party in a transaction to determine the identity of the other  
3849 party in the transaction. Authentication may be in one direction, or it may be bilateral.

3850 Given the sensitive nature of **access control** systems, it is important for a **PEP** to authenticate the  
3851 identity of the **PDP** to which it sends **decision requests**. Otherwise, there is a risk that an adversary  
3852 could provide false or invalid **authorization decisions**, leading to a policy violation.

3853 It is equally important for a **PDP** to authenticate the identity of the **PEP** and assess the level of trust to  
3854 determine what, if any, sensitive data should be passed. One should keep in mind that even simple  
3855 "Permit" or "Deny" responses could be exploited if an adversary were allowed to make unlimited requests  
3856 to a **PDP**.

3857 Many different techniques may be used to provide authentication, such as co-located code, a private  
3858 network, a VPN, or digital signatures. Authentication may also be performed as part of the  
3859 communication protocol used to exchange the **contexts**. In this case, authentication may be performed  
3860 either at the message level or at the session level.

### 3861 9.2.2 Policy administration

3862 If the contents of **policies** are exposed outside of the **access control** system, potential **subjects** may  
3863 use this information to determine how to gain unauthorized **access**.

3864 To prevent this threat, the repository used for the storage of **policies** may itself require **access control**.  
3865 In addition, the <Status> element should be used to return values of missing **attributes** only when  
3866 exposure of the identities of those **attributes** will not compromise security.

## 3867 9.2.3 Confidentiality

3868 Confidentiality mechanisms ensure that the contents of a message can be read only by the desired  
3869 recipients and not by anyone else who encounters the message while it is in transit. There are two areas  
3870 in which confidentiality should be considered: one is confidentiality during transmission; the other is  
3871 confidentiality within a <Policy> element.

### 3872 9.2.3.1 Communication confidentiality

3873 In some environments it is deemed good practice to treat all data within an **access control** system as  
3874 confidential. In other environments, **policies** may be made freely available for distribution, inspection,  
3875 and audit. The idea behind keeping **policy** information secret is to make it more difficult for an adversary  
3876 to know what steps might be sufficient to obtain unauthorized **access**. Regardless of the approach  
3877 chosen, the security of the **access control** system should not depend on the secrecy of the **policy**.

3878 Any security considerations related to transmitting or exchanging XACML <Policy> elements are  
3879 outside the scope of the XACML standard. While it is important to ensure that the integrity and  
3880 confidentiality of <Policy> elements is maintained when they are exchanged between two parties, it is  
3881 left to the implementers to determine the appropriate mechanisms for their environment.

3882 Communications confidentiality can be provided by a confidentiality mechanism, such as SSL. Using a  
3883 point-to-point scheme like SSL may lead to other vulnerabilities when one of the end-points is  
3884 compromised.

### 3885 9.2.3.2 Statement level confidentiality

3886 In some cases, an implementation may want to encrypt only parts of an XACML <Policy> element.

3887 The XML Encryption Syntax and Processing Candidate Recommendation from W3C can be used to  
3888 encrypt all or parts of an XML document. This specification is recommended for use with XACML.

3889 It should go without saying that if a repository is used to facilitate the communication of cleartext (i.e.,  
3890 unencrypted) **policy** between the **PAP** and **PDP**, then a secure repository should be used to store this  
3891 sensitive data.

## 3892 9.2.4 Policy integrity

3893 The XACML **policy** used by the **PDP** to evaluate the request **context** is the heart of the system.  
3894 Therefore, maintaining its integrity is essential. There are two aspects to maintaining the integrity of the  
3895 **policy**. One is to ensure that <Policy> elements have not been altered since they were originally  
3896 created by the **PAP**. The other is to ensure that <Policy> elements have not been inserted or deleted  
3897 from the set of **policies**.

3898 In many cases, both aspects can be achieved by ensuring the integrity of the actors and implementing  
3899 session-level mechanisms to secure the communication between actors. The selection of the appropriate  
3900 mechanisms is left to the implementers. However, when **policy** is distributed between organizations to  
3901 be acted on at a later time, or when the **policy** travels with the protected **resource**, it would be useful to  
3902 sign the **policy**. In these cases, the XML Signature Syntax and Processing standard from W3C is  
3903 recommended to be used with XACML.

3904 Digital signatures should only be used to ensure the integrity of the statements. Digital signatures should  
3905 not be used as a method of selecting or evaluating **policy**. That is, the **PDP** should not request a **policy**  
3906 based on who signed it or whether or not it has been signed (as such a basis for selection would, itself,  
3907 be a matter of policy). However, the **PDP** must verify that the key used to sign the **policy** is one  
3908 controlled by the purported **issuer** of the **policy**. The means to do this are dependent on the specific  
3909 signature technology chosen and are outside the scope of this document.

## 3910 9.2.5 Policy identifiers

3911 Since **policies** can be referenced by their identifiers, it is the responsibility of the **PAP** to ensure that  
3912 these are unique. Confusion between identifiers could lead to misidentification of the **applicable policy**.

3913 This specification is silent on whether a **PAP** must generate a new identifier when a **policy** is modified or  
3914 may use the same identifier in the modified **policy**. This is a matter of administrative practice. However,  
3915 care must be taken in either case. If the identifier is reused, there is a danger that other **policies** or  
3916 **policy sets** that reference it may be adversely affected. Conversely, if a new identifier is used, these  
3917 other **policies** may continue to use the prior **policy**, unless it is deleted. In either case the results may  
3918 not be what the **policy** administrator intends.

3919 If a **PDP** is provided with **policies** from distinct sources which might not be fully trusted, as in the use of  
3920 the administration profile [**XACMLAdmin**], there is a concern that someone could intentionally publish a  
3921 **policy** with an id which collides with another **policy**. This could cause **policy** references that point to the  
3922 wrong **policy**, and may cause other unintended consequences in an implementation which is predicated  
3923 upon having unique **policy** identifiers.

3924 If this issue is a concern it is RECOMMENDED that distinct **policy** issuers or sources are assigned  
3925 distinct namespaces for **policy** identifiers. One method is to make sure that the **policy** identifier begins  
3926 with a string which has been assigned to the particular **policy** issuer or source. The remainder of the  
3927 **policy** identifier is an issuer-specific unique part. For instance, Alice from Example Inc. could be assigned  
3928 the **policy** identifiers which begin with `http://example.com/xacml/policyId/alice/`. The **PDP** or another  
3929 trusted component can then verify that the authenticated source of the **policy** is Alice at Example Inc, or  
3930 otherwise reject the **policy**. Anyone else will be unable to publish **policies** with identifiers which collide  
3931 with the **policies** of Alice.

## 3932 9.2.6 Trust model

3933 Discussions of authentication, integrity and confidentiality safeguards necessarily assume an underlying  
3934 trust model: how can one actor come to believe that a given key is uniquely associated with a specific,  
3935 identified actor so that the key can be used to encrypt data for that actor or verify signatures (or other  
3936 integrity structures) from that actor? Many different types of trust models exist, including strict  
3937 hierarchies, distributed authorities, the Web, the bridge, and so on.

3938 It is worth considering the relationships between the various actors of the **access control** system in terms  
3939 of the interdependencies that do and do not exist.

- 3940 • None of the entities of the authorization system are dependent on the **PEP**. They may collect data  
3941 from it, (for example authentication data) but are responsible for verifying it themselves.
- 3942 • The correct operation of the system depends on the ability of the **PEP** to actually enforce **policy**  
3943 **decisions**.
- 3944 • The **PEP** depends on the **PDP** to correctly evaluate **policies**. This in turn implies that the **PDP** is  
3945 supplied with the correct inputs. Other than that, the **PDP** does not depend on the **PEP**.
- 3946 • The **PDP** depends on the **PAP** to supply appropriate **policies**. The **PAP** is not dependent on other  
3947 components.

## 3948 9.2.7 Privacy

3949 It is important to be aware that any transactions that occur with respect to **access control** may reveal  
3950 private information about the actors. For example, if an XACML **policy** states that certain data may only  
3951 be read by **subjects** with "Gold Card Member" status, then any transaction in which a **subject** is  
3952 permitted **access** to that data leaks information to an adversary about the **subject's** status. Privacy  
3953 considerations may therefore lead to encryption and/or to **access control** requirements surrounding the  
3954 enforcement of XACML **policy** instances themselves: confidentiality-protected channels for the  
3955 request/response protocol messages, protection of **subject attributes** in storage and in transit, and so  
3956 on.

3957 Selection and use of privacy mechanisms appropriate to a given environment are outside the scope of  
3958 XACML. The **decision** regarding whether, how, and when to deploy such mechanisms is left to the  
3959 implementers associated with the environment.

3960 **9.3 Unicode security issues**

3961 There are many security considerations related to use of Unicode. An XACML implementation SHOULD  
3962 follow the advice given in the relevant version of [UTR36].

3963 **9.4 Identifier equality**

3964 Section 7.20 defines the identifier equality operation for XACML. This definition of equality does not do  
3965 any kind of canonicalization or escaping of characters. The identifiers defined in the XACML specification  
3966 have been selected to not include any ambiguity regarding these aspects. It is RECOMMENDED that  
3967 identifiers defined by extensions also do not introduce any identifiers which might be mistaken for being  
3968 subject to processing, like for instance URL character encoding using "%".

3969 **10 Conformance**

3970 **10.1 Introduction**

3971 The XACML specification addresses the following aspect of conformance:

3972 The XACML specification defines a number of functions, etc. that have somewhat special applications,  
3973 therefore they are not required to be implemented in an implementation that claims to conform with the  
3974 OASIS standard.

3975 **10.2 Conformance tables**

3976 This section lists those portions of the specification that **MUST** be included in an implementation of a **PDP**  
3977 that claims to conform to XACML v3.0. A set of test cases has been created to assist in this process.  
3978 These test cases can be located from the OASIS XACML TC Web page. The site hosting the test cases  
3979 contains a full description of the test cases and how to execute them.

3980 Note: "M" means mandatory-to-implement. "O" means optional.

3981 The implementation **MUST** follow sections 5, 6, 7, Appendix A, Appendix B and Appendix C where they  
3982 apply to implemented items in the following tables.

3983 **10.2.1 Schema elements**

3984 The implementation **MUST** support those schema elements that are marked "M".

Element name	M/O
xacml:Advice	M
xacml:AdviceExpression	M
xacml:AdviceExpressions	M
xacml:AllOf	M
xacml:AnyOf	M
xacml:Apply	M
xacml:AssociatedAdvice	M
xacml:Attribute	M
xacml:AttributeAssignment	M
xacml:AttributeAssignmentExpression	M
xacml:AttributeDesignator	M
xacml:Attributes	M
xacml:AttributeSelector	O
xacml:AttributesReference	O
xacml:AttributeValue	M
xacml:CombinerParameter	O
xacml:CombinerParameters	O
xacml:Condition	M
xacml:Content	O
xacml:Decision	M
xacml:Description	M
xacml:Expression	M
xacml:Function	M
xacml:Match	M
xacml:MissingAttributeDetail	M
xacml:MultiRequests	O
xacml:Obligation	M
xacml:ObligationExpression	M
xacml:ObligationExpressions	M
xacml:Obligations	M

xacml:Policy	M
xacml:PolicyCombinerParameters	O
xacml:PolicyDefaults	O
xacml:PolicyIdentifierList	O
xacml:PolicyIdReference	M
xacml:PolicyIssuer	O
xacml:PolicySet	M
xacml:PolicySetDefaults	O
xacml:PolicySetIdReference	M
xacml:Request	M
xacml:RequestDefaults	O
xacml:RequestReference	O
xacml:Response	M
xacml:Result	M
xacml:Rule	M
xacml:RuleCombinerParameters	O
xacml:Status	M
xacml:StatusCode	M
xacml:StatusDetail	O
xacml:StatusMessage	O
xacml:Target	M
xacml:VariableDefinition	M
xacml:VariableReference	M
xacml:XPathVersion	O

3985 **10.2 Identifier Prefixes**

3986 The following identifier prefixes are reserved by XACML.

Identifier
urn:oasis:names:tc:xacml:3.0
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

3987 **10.2.3 Algorithms**

3988 The implementation MUST include the *rule-* and *policy-combining algorithms* associated with the  
3989 following identifiers that are marked "M".

Algorithm	M/O
urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides	M
urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides	M
urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides	M
urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides	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-applicable	M
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-	M



applicable	
urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides	M
urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-overrides	M
urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-overrides	M
urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-overrides	M
urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit	M
urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit	M
urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-unless-deny	M
urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-unless-deny	M

3990 **10.2.4 Status Codes**

3991 Implementation support for the <StatusCode> element is optional, but if the element is supported, then  
3992 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

3993 **10.2.5 Attributes**

3994 The implementation MUST support the **attributes** associated with the following identifiers as specified by  
3995 XACML. If values for these **attributes** are not present in the **decision request**, then their values MUST  
3996 be supplied by the **context handler**. So, unlike most other **attributes**, their semantics are not  
3997 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

3998 **10.2.6 Identifiers**

3999 The implementation MUST use the **attributes** associated with the following identifiers in the way XACML  
4000 has defined. This requirement pertains primarily to implementations of a **PAP** or **PEP** that uses XACML,  
4001 since the semantics of the **attributes** are transparent to the **PDP**.

Identifier	M/O
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name	O
urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address	O
urn:oasis:names:tc:xacml:1.0:subject:authentication-method	O
urn:oasis:names:tc:xacml:1.0:subject:authentication-time	O
urn:oasis:names:tc:xacml:1.0:subject:key-info	O
urn:oasis:names:tc:xacml:1.0:subject:request-time	O
urn:oasis:names:tc:xacml:1.0:subject:session-start-time	O
urn:oasis:names:tc:xacml:1.0:subject:subject-id	O
urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier	O
urn:oasis:names:tc:xacml:1.0:subject-category:access-subject	M
urn:oasis:names:tc:xacml:1.0:subject-category:codebase	O

urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject	O
urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject	O
urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine	O
urn:oasis:names:tc:xacml:1.0:resource:resource-location	O
urn:oasis:names:tc:xacml:1.0:resource:resource-id	M
urn:oasis:names:tc:xacml:1.0:resource:simple-file-name	O
urn:oasis:names:tc:xacml:1.0:action:action-id	O
urn:oasis:names:tc:xacml:1.0:action:implied-action	O

4002 **10.2.7 Data-types**

4003 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	M
http://www.w3.org/2001/XMLSchema#date	M
http://www.w3.org/2001/XMLSchema#dateTime	M
http://www.w3.org/2001/XMLSchema#dayTimeDuration	M
http://www.w3.org/2001/XMLSchema#yearMonthDuration	M
http://www.w3.org/2001/XMLSchema#anyURI	M
http://www.w3.org/2001/XMLSchema#hexBinary	M
http://www.w3.org/2001/XMLSchema#base64Binary	M
urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name	M
urn:oasis:names:tc:xacml:1.0:data-type:x500Name	M
urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression	O
urn:oasis:names:tc:xacml:2.0:data-type:ipAddress	M
urn:oasis:names:tc:xacml:2.0:data-type:dnsName	M

4004 **10.2.8 Functions**

4005 The implementation MUST properly process those functions associated with the identifiers marked with  
4006 an "M".

Function	M/O
urn:oasis:names:tc:xacml:1.0:function:string-equal	M
urn:oasis:names:tc:xacml:1.0:function:boolean-equal	M
urn:oasis:names:tc:xacml:1.0:function:integer-equal	M
urn:oasis:names:tc:xacml:1.0:function:double-equal	M
urn:oasis:names:tc:xacml:1.0:function:date-equal	M
urn:oasis:names:tc:xacml:1.0:function:time-equal	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-equal	M
urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-equal	M
urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-equal	M
urn:oasis:names:tc:xacml:3.0:function:string-equal-ignore-case	M
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	M
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	M

urn:oasis:names:tc:xacml:1.0:function:double-multiply	M
urn:oasis:names:tc:xacml:1.0:function:integer-divide	M
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	M
urn:oasis:names:tc:xacml:1.0:function:round	M
urn:oasis:names:tc:xacml:1.0:function:floor	M
urn:oasis:names:tc:xacml:1.0:function:string-normalize-space	M
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	M
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	M
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	M
urn:oasis:names:tc:xacml:1.0:function:double-less-than	M
urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal	M
urn:oasis:names:tc:xacml:3.0:function:dateTime-add-dayTimeDuration	M
urn:oasis:names:tc:xacml:3.0:function:dateTime-add-yearMonthDuration	M
urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-dayTimeDuration	M
urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-yearMonthDuration	M
urn:oasis:names:tc:xacml:3.0:function:date-add-yearMonthDuration	M
urn:oasis:names:tc:xacml:3.0:function:date-subtract-yearMonthDuration	M
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	M
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	M
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	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than	M
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	M
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	M
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	M
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	M
urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:integer-bag-size	M

urn:oasis:names:tc:xacml:1.0:function:integer-is-in	M
urn:oasis:names:tc:xacml:1.0:function:integer-bag	M
urn:oasis:names:tc:xacml:1.0:function:double-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:double-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:double-is-in	M
urn:oasis:names:tc:xacml:1.0:function:double-bag	M
urn:oasis:names:tc:xacml:1.0:function:time-one-and-only	M
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	M
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	M
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	M
urn:oasis:names:tc:xacml:1.0:function:anyURI-is-in	M
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	M
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	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-is-in	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-bag	M
urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-one-and-only	M
urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-bag-size	M
urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-is-in	M
urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-bag	M
urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-one-and-only	M
urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-bag-size	M
urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-is-in	M
urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-bag	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-one-and-only	M
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	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-bag-size	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-is-in	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-bag	M
urn:oasis:names:tc:xacml:2.0:function:ipAddress-one-and-only	M
urn:oasis:names:tc:xacml:2.0:function:ipAddress-bag-size	M
urn:oasis:names:tc:xacml:2.0:function:ipAddress-bag	M
urn:oasis:names:tc:xacml:2.0:function:dnsName-one-and-only	M
urn:oasis:names:tc:xacml:2.0:function:dnsName-bag-size	M
urn:oasis:names:tc:xacml:2.0:function:dnsName-bag	M
urn:oasis:names:tc:xacml:2.0:function:string-concatenate	M
urn:oasis:names:tc:xacml:3.0:function:boolean-from-string	M
urn:oasis:names:tc:xacml:3.0:function:string-from-boolean	M
urn:oasis:names:tc:xacml:3.0:function:integer-from-string	M
urn:oasis:names:tc:xacml:3.0:function:string-from-integer	M
urn:oasis:names:tc:xacml:3.0:function:double-from-string	M

urn:oasis:names:tc:xacml:3.0:function:string-from-double	M
urn:oasis:names:tc:xacml:3.0:function:time-from-string	M
urn:oasis:names:tc:xacml:3.0:function:string-from-time	M
urn:oasis:names:tc:xacml:3.0:function:date-from-string	M
urn:oasis:names:tc:xacml:3.0:function:string-from-date	M
urn:oasis:names:tc:xacml:3.0:function:dateTime-from-string	M
urn:oasis:names:tc:xacml:3.0:function:string-from-dateTime	M
urn:oasis:names:tc:xacml:3.0:function:anyURI-from-string	M
urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI	M
urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-from-string	M
urn:oasis:names:tc:xacml:3.0:function:string-from-dayTimeDuration	M
urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-from-string	M
urn:oasis:names:tc:xacml:3.0:function:string-from-yearMonthDuration	M
urn:oasis:names:tc:xacml:3.0:function:x500Name-from-string	M
urn:oasis:names:tc:xacml:3.0:function:string-from-x500Name	M
urn:oasis:names:tc:xacml:3.0:function:rfc822Name-from-string	M
urn:oasis:names:tc:xacml:3.0:function:string-from-rfc822Name	M
urn:oasis:names:tc:xacml:3.0:function:ipAddress-from-string	M
urn:oasis:names:tc:xacml:3.0:function:string-from-ipAddress	M
urn:oasis:names:tc:xacml:3.0:function:dnsName-from-string	M
urn:oasis:names:tc:xacml:3.0:function:string-from-dnsName	M
urn:oasis:names:tc:xacml:3.0:function:string-starts-with	M
urn:oasis:names:tc:xacml:3.0:function:anyURI-starts-with	M
urn:oasis:names:tc:xacml:3.0:function:string-ends-with	M
urn:oasis:names:tc:xacml:3.0:function:anyURI-ends-with	M
urn:oasis:names:tc:xacml:3.0:function:string-contains	M
urn:oasis:names:tc:xacml:3.0:function:anyURI-contains	M
urn:oasis:names:tc:xacml:3.0:function:string-substring	M
urn:oasis:names:tc:xacml:3.0:function:anyURI-substring	M
urn:oasis:names:tc:xacml:3.0:function:any-of	M
urn:oasis:names:tc:xacml:3.0:function:all-of	M
urn:oasis:names:tc:xacml:3.0: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	M
urn:oasis:names:tc:xacml:3.0:function:map	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-match	M
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	M
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	M
urn:oasis:names:tc:xacml:3.0:function:xpath-node-count	O
urn:oasis:names:tc:xacml:3.0:function:xpath-node-equal	O
urn:oasis:names:tc:xacml:3.0:function:xpath-node-match	O
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	M
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	M
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	M
urn:oasis:names:tc:xacml:1.0:function:integer-intersection	M

urn:oasis:names:tc:xacml:1.0:function:integer-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:integer-union	M
urn:oasis:names:tc:xacml:1.0:function:integer-subset	M
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	M
urn:oasis:names:tc:xacml:1.0:function:double-subset	M
urn:oasis:names:tc:xacml:1.0:function:double-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:time-intersection	M
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	M
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	M
urn:oasis:names:tc:xacml:1.0:function:date-subset	M
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	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-union	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-subset	M
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	M
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	M
urn:oasis:names:tc:xacml:1.0:function:hexBinary-set-equals	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-intersection	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-union	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-subset	M
urn:oasis:names:tc:xacml:1.0:function:base64Binary-set-equals	M
urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-intersection	M
urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-at-least-one-member-of	M
urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-union	M
urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-subset	M
urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-set-equals	M
urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-intersection	M
urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-at-least-one-member-of	M
urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-union	M
urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-subset	M
urn:oasis:names:tc:xacml:3.0: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	M
urn:oasis:names:tc:xacml:1.0:function:x500Name-subset	M
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	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-union	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-subset	M
urn:oasis:names:tc:xacml:1.0:function:rfc822Name-set-equals	M
urn:oasis:names:tc:xacml:3.0:function:access-permitted	O

4007 **10.2.9 Identifiers planned for future deprecation**

4008 These identifiers are associated with previous versions of XACML and newer alternatives exist in XACML  
4009 3.0. They are planned to be deprecated at some unspecified point in the future. It is RECOMMENDED  
4010 that these identifiers not be used in new policies and requests.

4011 The implementation MUST properly process those features associated with the identifiers marked with an  
4012 "M".

Function	M/O
urn:oasis:names:tc:xacml:1.0:function:xpath-node-count	O
urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal	O
urn:oasis:names:tc:xacml:1.0:function:xpath-node-match	O
urn:oasis:names:tc:xacml:2.0:function:uri-string-concatenate	M
http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration	M
http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-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	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration	M
urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-yearMonthDuration	M
urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration	M
urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration	M
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	M
urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides	M
urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides	M
urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides	M
urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-overrides	M
urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-overrides	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-intersection	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-at-least-one-member-of	M
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	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-intersection	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-at-least-one-member-of	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-union	M
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:dayTimeDuration-one-and-only	M
urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-bag-size	M

urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-is-in	M
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	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-is-in	M
urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-bag	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:map	M

4013



## 4014 Appendix A. Data-types and functions (normative)

### 4015 A.1 Introduction

4016 This section specifies the data-types and functions used in XACML to create *predicates* for *conditions*  
4017 and *target* matches.

4018 This specification combines the various standards set forth by IEEE and ANSI for string representation of  
4019 numeric values, as well as the evaluation of arithmetic functions. It describes the primitive data-types and  
4020 *bags*. The standard functions are named and their operational semantics are described.

### 4021 A.2 Data-types

4022 Although XML instances represent all data-types as strings, an XACML *PDP* must operate on types of  
4023 data that, while they have string representations, are not just strings. Types such as Boolean, integer,  
4024 and double MUST be converted from their XML string representations to values that can be compared  
4025 with values in their domain of discourse, such as numbers. The following primitive data-types are  
4026 specified for use with XACML and have explicit data representations:

- 4027 • <http://www.w3.org/2001/XMLSchema#string>
- 4028 • <http://www.w3.org/2001/XMLSchema#boolean>
- 4029 • <http://www.w3.org/2001/XMLSchema#integer>
- 4030 • <http://www.w3.org/2001/XMLSchema#double>
- 4031 • <http://www.w3.org/2001/XMLSchema#time>
- 4032 • <http://www.w3.org/2001/XMLSchema#date>
- 4033 • <http://www.w3.org/2001/XMLSchema#dateTime>
- 4034 • <http://www.w3.org/2001/XMLSchema#anyURI>
- 4035 • <http://www.w3.org/2001/XMLSchema#hexBinary>
- 4036 • <http://www.w3.org/2001/XMLSchema#base64Binary>
- 4037 • <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- 4038 • <http://www.w3.org/2001/XMLSchema#yearMonthDuration>
- 4039 • <urn:oasis:names:tc:xacml:1.0:data-type:x500Name>
- 4040 • <urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name>
- 4041 • <urn:oasis:names:tc:xacml:2.0:data-type:ipAddress>
- 4042 • <urn:oasis:names:tc:xacml:2.0:data-type:dnsName>
- 4043 • <urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression>

4044 For the sake of improved interoperability, it is RECOMMENDED that all time references be in UTC time.

4045 An XACML *PDP* SHALL be capable of converting string representations into various primitive data-types.

4046 For doubles, XACML SHALL use the conversions described in [IEEE754].

4047 XACML defines four data-types representing identifiers for *subjects* or *resources*; these are:

4048 "urn:oasis:names:tc:xacml:1.0:data-type:x500Name",

4049 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"

4050 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" and

4051 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName"

4052 These types appear in several standard applications, such as TLS/SSL and electronic mail.

#### 4053 X.500 directory name

4054 The "urn:oasis:names:tc:xacml:1.0:data-type:x500Name" primitive type represents an ITU-T Rec.  
4055 X.520 Distinguished Name. The valid syntax for such a name is described in IETF RFC 2253  
4056 "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished  
4057 Names".

#### 4058 RFC 822 name

4059 The "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" primitive type represents an electronic  
4060 mail address. The valid syntax for such a name is described in IETF RFC 2821, Section 4.1.2,  
4061 Command Argument Syntax, under the term "Mailbox".

#### 4062 IP address

4063 The "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" primitive type represents an IPv4 or IPv6  
4064 network address, with optional mask and optional port or port range. The syntax SHALL be:

4065 ipAddress = address [ "/" mask ] [ ":" [ portrange ] ]

4066 For an IPv4 address, the address and mask are formatted in accordance with the syntax for a  
4067 "host" in IETF RFC 2396 "Uniform Resource Identifiers (URI): Generic Syntax", section 3.2.

4068 For an IPv6 address, the address and mask are formatted in accordance with the syntax for an  
4069 "ipv6reference" in IETF RFC 2732 "Format for Literal IPv6 Addresses in URL's". (Note that an  
4070 IPv6 address or mask, in this syntax, is enclosed in literal "[" "]" brackets.)

#### 4071 DNS name

4072 The "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" primitive type represents a Domain  
4073 Name Service (DNS) host name, with optional port or port range. The syntax SHALL be:

4074 dnsName = hostname [ ":" portrange ]

4075 The hostname is formatted in accordance with IETF RFC 2396 "Uniform Resource Identifiers  
4076 (URI): Generic Syntax", section 3.2, except that a wildcard "\*" may be used in the left-most  
4077 component of the hostname to indicate "any subdomain" under the domain specified to its right.

4078 For both the "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" and  
4079 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" data-types, the port or port range syntax  
4080 SHALL be

4081 portrange = portnumber | "-"portnumber | portnumber "-" [ portnumber ]

4082 where "portnumber" is a decimal port number. If the port number is of the form "-x", where "x" is  
4083 a port number, then the range is all ports numbered "x" and below. If the port number is of the  
4084 form "x-", then the range is all ports numbered "x" and above. [This syntax is taken from the Java  
4085 SocketPermission.]

#### 4086 XPath expression

4087 The "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" primitive type represents an  
4088 XPath expression over the XML in a <Content> element. The syntax is defined by the XPath  
4089 W3C recommendation. The content of this data type also includes the context in which  
4090 namespaces prefixes in the expression are resolved, which distinguishes it from a plain string and  
4091 the XACML *attribute* category of the <Content> element to which it applies. When the value is  
4092 encoded in an <AttributeValue> element, the namespace context is given by the [\[in-scope](#)  
4093 [namespaces\]](#) (see [\[INFOSET1\]](#) of the <AttributeValue> element, and an XML attribute called  
4094 XPathCategory gives the category of the <Content> element where the expression applies.

4095 The XPath expression MUST be evaluated in a context which is equivalent of a stand alone XML  
4096 document with the only child of the <Content> element as the document element. Namespace  
4097 declarations which are not "visibly utilized", as defined by [\[exc-c14n\]](#), MAY not be present and  
4098 MUST NOT be utilized by the XPath expression. The context node of the XPath expression is the  
4099 document node of this stand alone document.

Formatted: Default Paragraph Font

Formatted: Default Paragraph Font

Formatted: Default Paragraph Font

## 4100 A.3 Functions

4101 XACML specifies the following functions. Unless otherwise specified, if an argument of one of these  
4102 functions were to evaluate to "Indeterminate", then the function SHALL be set to "Indeterminate".

4103 Note that in each case an implementation is conformant as long as it produces the same result as is  
4104 specified here, regardless of how and in what order the implementation behaves internally.

### 4105 A.3.1 Equality predicates

4106 The following functions are the equality functions for the various primitive types. Each function for a  
4107 particular data-type follows a specified standard convention for that data-type.

- 4108 • urn:oasis:names:tc:xacml:1.0:function:string-equal  
4109     This function SHALL take two arguments of data-type  
4110     "http://www.w3.org/2001/XMLSchema#string" and SHALL return an  
4111     "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if  
4112     the value of both of its arguments are of equal length and each string is determined to be equal.  
4113     Otherwise, it SHALL return "False". The comparison SHALL use Unicode codepoint collation, as  
4114     defined for the identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [XF].
- 4115 • urn:oasis:names:tc:xacml:3.0:function:string-equal-ignore-case  
4116     This function SHALL take two arguments of data-type  
4117     "http://www.w3.org/2001/XMLSchema#string" and SHALL return an  
4118     "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be "True" if and only if the  
4119     two strings are equal as defined by urn:oasis:names:tc:xacml:1.0:function:string-equal after they  
4120     have both been converted to lower case with urn:oasis:names:tc:xacml:1.0:function:string-  
4121     normalize-to-lower-case.
- 4122 • urn:oasis:names:tc:xacml:1.0:function:boolean-equal  
4123     This function SHALL take two arguments of data-type  
4124     "http://www.w3.org/2001/XMLSchema#boolean" and SHALL return an  
4125     "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if  
4126     the arguments are equal. Otherwise, it SHALL return "False".
- 4127 • urn:oasis:names:tc:xacml:1.0:function:integer-equal  
4128     This function SHALL take two arguments of data-type  
4129     "http://www.w3.org/2001/XMLSchema#integer" and SHALL return an  
4130     "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if  
4131     the two arguments represent the same number.
- 4132 • urn:oasis:names:tc:xacml:1.0:function:double-equal  
4133     This function SHALL take two arguments of data-type  
4134     "http://www.w3.org/2001/XMLSchema#double" and SHALL return an  
4135     "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation on doubles  
4136     according to IEEE 754 [IEEE754].
- 4137 • urn:oasis:names:tc:xacml:1.0:function:date-equal  
4138     This function SHALL take two arguments of data-type  
4139     "http://www.w3.org/2001/XMLSchema#date" and SHALL return an  
4140     "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to  
4141     the "op:date-equal" function [XF] Section 10.4.9.
- 4142 • urn:oasis:names:tc:xacml:1.0:function:time-equal  
4143     This function SHALL take two arguments of data-type  
4144     "http://www.w3.org/2001/XMLSchema#time" and SHALL return an  
4145     "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to  
4146     the "op:time-equal" function [XF] Section 10.4.12.

- 4147 • urn:oasis:names:tc:xacml:1.0:function:dateTime-equal  
4148 This function SHALL take two arguments of data-type  
4149 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an  
4150 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to  
4151 the "op:dateTime-equal" function [XF] Section 10.4.6.
- 4152 • urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-equal  
4153 This function SHALL take two arguments of data-type  
4154 "http://www.w3.org/2001/XMLSchema#dayTimeDuration" and SHALL return an  
4155 "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation  
4156 according to the "op:duration-equal" function [XF] Section 10.4.5. Note that the lexical  
4157 representation of each argument MUST be converted to a value expressed in fractional seconds  
4158 [XF] Section 10.3.2.
- 4159 • urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-equal  
4160 This function SHALL take two arguments of data-type  
4161 "http://www.w3.org/2001/XMLSchema#yearMonthDuration" and SHALL return an  
4162 "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation  
4163 according to the "op:duration-equal" function [XF] Section 10.4.5. Note that the lexical  
4164 representation of each argument MUST be converted to a value expressed in fractional seconds  
4165 [XF] Section 10.3.2.
- 4166 • urn:oasis:names:tc:xacml:1.0:function:anyURI-equal  
4167 This function SHALL take two arguments of data-type  
4168 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return an  
4169 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL convert the arguments to  
4170 strings with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI and return "True" if and  
4171 only if the values of the two arguments are equal on a codepoint-by-codepoint basis.
- 4172 • urn:oasis:names:tc:xacml:1.0:function:x500Name-equal  
4173 This function SHALL take two arguments of "urn:oasis:names:tc:xacml:1.0:data-type:x500Name"  
4174 and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if  
4175 and only if each Relative Distinguished Name (RDN) in the two arguments matches. Otherwise,  
4176 it SHALL return "False". Two RDNs shall be said to match if and only if the result of the following  
4177 operations is "True".
- 4178 1. Normalize the two arguments according to IETF RFC 2253 "Lightweight Directory Access  
4179 Protocol (v3): UTF-8 String Representation of Distinguished Names".
  - 4180 2. If any RDN contains multiple attributeTypeAndValue pairs, re-order the Attribute  
4181 ValuePairs in that RDN in ascending order when compared as octet strings (described in  
4182 ITU-T Rec. X.690 (1997 E) Section 11.6 "Set-of components").
  - 4183 3. Compare RDNs using the rules in IETF RFC 3280 "Internet X.509 Public Key  
4184 Infrastructure Certificate and Certificate Revocation List (CRL) Profile", Section 4.1.2.4  
4185 "Issuer".
- 4186 • urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal  
4187 This function SHALL take two arguments of data-type "urn:oasis:names:tc:xacml:1.0:data-  
4188 type:rfc822Name" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It  
4189 SHALL return "True" if and only if the two arguments are equal. Otherwise, it SHALL return  
4190 "False". An RFC822 name consists of a local-part followed by "@" followed by a domain-part.  
4191 The local-part is case-sensitive, while the domain-part (which is usually a DNS host name) is not  
4192 case-sensitive. Perform the following operations:
- 4193 1. Normalize the domain-part of each argument to lower case
  - 4194 2. Compare the expressions by applying the function  
4195 "urn:oasis:names:tc:xacml:1.0:function:string-equal" to the normalized arguments.

- 4196 • urn:oasis:names:tc:xacml:1.0:function:hexBinary-equal
- 4197     This function SHALL take two arguments of data-type
- 4198     "http://www.w3.org/2001/XMLSchema#hexBinary" and SHALL return an
- 4199     "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet sequences
- 4200     represented by the value of both arguments have equal length and are equal in a conjunctive,
- 4201     point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function.
- 4202     Otherwise, it SHALL return "False". The conversion from the string representation to an octet
- 4203     sequence SHALL be as specified in **[XS]** Section 3.2.15.
  
- 4204 • urn:oasis:names:tc:xacml:1.0:function:base64Binary-equal
- 4205     This function SHALL take two arguments of data-type
- 4206     "http://www.w3.org/2001/XMLSchema#base64Binary" and SHALL return an
- 4207     "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet sequences
- 4208     represented by the value of both arguments have equal length and are equal in a conjunctive,
- 4209     point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function.
- 4210     Otherwise, it SHALL return "False". The conversion from the string representation to an octet
- 4211     sequence SHALL be as specified in **[XS]** Section 3.2.16.

### 4212 **A.3.2 Arithmetic functions**

4213 All of the following functions SHALL take two arguments of the specified data-type, integer, or double,  
 4214 and SHALL return an element of integer or double data-type, respectively. However, the "add" and  
 4215 "multiply" functions MAY take more than two arguments. Each function evaluation operating on doubles  
 4216 SHALL proceed as specified by their logical counterparts in IEEE 754 **[IEEE754]**. For all of these  
 4217 functions, if any argument is "Indeterminate", then the function SHALL evaluate to "Indeterminate". In the  
 4218 case of the divide functions, if the divisor is zero, then the function SHALL evaluate to "Indeterminate".

- 4219 • urn:oasis:names:tc:xacml:1.0:function:integer-add
- 4220     This function MUST accept two or more arguments.
- 4221 • urn:oasis:names:tc:xacml:1.0:function:double-add
- 4222     This function MUST accept two or more arguments.
- 4223 • urn:oasis:names:tc:xacml:1.0:function:integer-subtract
- 4224     The result is the second argument subtracted from the first argument.
- 4225 • urn:oasis:names:tc:xacml:1.0:function:double-subtract
- 4226     The result is the second argument subtracted from the first argument.
- 4227 • urn:oasis:names:tc:xacml:1.0:function:integer-multiply
- 4228     This function MUST accept two or more arguments.
- 4229 • urn:oasis:names:tc:xacml:1.0:function:double-multiply
- 4230     This function MUST accept two or more arguments.
- 4231 • urn:oasis:names:tc:xacml:1.0:function:integer-divide
- 4232     The result is the first argument divided by the second argument.
- 4233 • urn:oasis:names:tc:xacml:1.0:function:double-divide
- 4234     The result is the first argument divided by the second argument.
- 4235 • urn:oasis:names:tc:xacml:1.0:function:integer-mod
- 4236     The result is remainder of the first argument divided by the second argument.
- 4237 The following functions SHALL take a single argument of the specified data-type. The round and floor
- 4238 functions SHALL take a single argument of data-type "http://www.w3.org/2001/XMLSchema#double" and
- 4239 return a value of the data-type "http://www.w3.org/2001/XMLSchema#double".
- 4240 • urn:oasis:names:tc:xacml:1.0:function:integer-abs

- 4241 • urn:oasis:names:tc:xacml:1.0:function:double-abs
- 4242 • urn:oasis:names:tc:xacml:1.0:function:round
- 4243 • urn:oasis:names:tc:xacml:1.0:function:floor

### 4244 **A.3.3 String conversion functions**

4245 The following functions convert between values of the data-type  
4246 "http://www.w3.org/2001/XMLSchema#string" primitive types.

- 4247 • urn:oasis:names:tc:xacml:1.0:function:string-normalize-space
  - 4248 This function SHALL take one argument of data-type
  - 4249 "http://www.w3.org/2001/XMLSchema#string" and SHALL normalize the value by stripping off all
  - 4250 leading and trailing white space characters. The whitespace characters are defined in the
  - 4251 metasympol S (Production 3) of [XML].
- 4252 • urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case
  - 4253 This function SHALL take one argument of data-type
  - 4254 "http://www.w3.org/2001/XMLSchema#string" and SHALL normalize the value by converting each
  - 4255 upper case character to its lower case equivalent. Case mapping shall be done as specified for
  - 4256 the fn:lower-case function in [XF] with no tailoring for particular languages or environments.

### 4257 **A.3.4 Numeric data-type conversion functions**

4258 The following functions convert between the data-type "http://www.w3.org/2001/XMLSchema#integer"  
4259 and "http://www.w3.org/2001/XMLSchema#double" primitive types.

- 4260 • urn:oasis:names:tc:xacml:1.0:function:double-to-integer
  - 4261 This function SHALL take one argument of data-type
  - 4262 "http://www.w3.org/2001/XMLSchema#double" and SHALL truncate its numeric value to a whole
  - 4263 number and return an element of data-type "http://www.w3.org/2001/XMLSchema#integer".
- 4264 • urn:oasis:names:tc:xacml:1.0:function:integer-to-double
  - 4265 This function SHALL take one argument of data-type
  - 4266 "http://www.w3.org/2001/XMLSchema#integer" and SHALL promote its value to an element of
  - 4267 data-type "http://www.w3.org/2001/XMLSchema#double" with the same numeric value. If the
  - 4268 integer argument is outside the range which can be represented by a double, the result SHALL
  - 4269 be indeterminate, with the status code "urn:oasis:names:tc:xacml:1.0:status:processing-error".

### 4270 **A.3.5 Logical functions**

4271 This section contains the specification for logical functions that operate on arguments of data-type  
4272 "http://www.w3.org/2001/XMLSchema#boolean".

- 4273 • urn:oasis:names:tc:xacml:1.0:function:or
  - 4274 This function SHALL return "False" if it has no arguments and SHALL return "True" if at least one
  - 4275 of its arguments evaluates to "True". The order of evaluation SHALL be from first argument to
  - 4276 last. The evaluation SHALL stop with a result of "True" if any argument evaluates to "True",
  - 4277 leaving the rest of the arguments unevaluated.
- 4278 • urn:oasis:names:tc:xacml:1.0:function:and
  - 4279 This function SHALL return "True" if it has no arguments and SHALL return "False" if one of its
  - 4280 arguments evaluates to "False". The order of evaluation SHALL be from first argument to last.
  - 4281 The evaluation SHALL stop with a result of "False" if any argument evaluates to "False", leaving
  - 4282 the rest of the arguments unevaluated.
- 4283 • urn:oasis:names:tc:xacml:1.0:function:n-of
  - 4284 The first argument to this function SHALL be of data-type
  - 4285 "http://www.w3.org/2001/XMLSchema#integer". The remaining arguments SHALL be of data-type

4286 http://www.w3.org/2001/XMLSchema#boolean. The first argument specifies the minimum  
4287 number of the remaining arguments that MUST evaluate to "True" for the expression to be  
4288 considered "True". If the first argument is 0, the result SHALL be "True". If the number of  
4289 arguments after the first one is less than the value of the first argument, then the expression  
4290 SHALL result in "Indeterminate". The order of evaluation SHALL be: first evaluate the integer  
4291 value, and then evaluate each subsequent argument. The evaluation SHALL stop and return  
4292 "True" if the specified number of arguments evaluate to "True". The evaluation of arguments  
4293 SHALL stop if it is determined that evaluating the remaining arguments will not satisfy the  
4294 requirement.

4295 • urn:oasis:names:tc:xacml:1.0:function:not

4296 This function SHALL take one argument of data-type  
4297 "http://www.w3.org/2001/XMLSchema#boolean". If the argument evaluates to "True", then the  
4298 result of the expression SHALL be "False". If the argument evaluates to "False", then the result  
4299 of the expression SHALL be "True".

4300 Note: When evaluating and, or, or n-of, it may not be necessary to attempt a full evaluation of each  
4301 argument in order to determine whether the evaluation of the argument would result in "Indeterminate".  
4302 Analysis of the argument regarding the availability of its **attributes**, or other analysis regarding errors,  
4303 such as "divide-by-zero", may render the argument error free. Such arguments occurring in the  
4304 expression in a position after the evaluation is stated to stop need not be processed.

Deleted: MAY NOT

### 4305 A.3.6 Numeric comparison functions

4306 These functions form a minimal set for comparing two numbers, yielding a Boolean result. For doubles  
4307 they SHALL comply with the rules governed by IEEE 754 [IEEE754].

- 4308 • urn:oasis:names:tc:xacml:1.0:function:integer-greater-than
- 4309 • urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal
- 4310 • urn:oasis:names:tc:xacml:1.0:function:integer-less-than
- 4311 • urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal
- 4312 • urn:oasis:names:tc:xacml:1.0:function:double-greater-than
- 4313 • urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal
- 4314 • urn:oasis:names:tc:xacml:1.0:function:double-less-than
- 4315 • urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal

### 4316 A.3.7 Date and time arithmetic functions

4317 These functions perform arithmetic operations with date and time.

4318 • urn:oasis:names:tc:xacml:3.0:function:dateTime-add-dayTimeDuration

4319 This function SHALL take two arguments, the first SHALL be of data-type  
4320 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be of data-type  
4321 "http://www.w3.org/2001/XMLSchema#dayTimeDuration". It SHALL return a result of  
4322 "http://www.w3.org/2001/XMLSchema#dateTime". This function SHALL return the value by  
4323 adding the second argument to the first argument according to the specification of adding  
4324 durations to date and time [XS] Appendix E.

4325 • urn:oasis:names:tc:xacml:3.0:function:dateTime-add-yearMonthDuration

4326 This function SHALL take two arguments, the first SHALL be a  
4327 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a  
4328 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of  
4329 "http://www.w3.org/2001/XMLSchema#dateTime". This function SHALL return the value by  
4330 adding the second argument to the first argument according to the specification of adding  
4331 durations to date and time [XS] Appendix E.

- 4333 • urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-dayTimeDuration
- 4334 This function SHALL take two arguments, the first SHALL be a  
 4335 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a  
 4336 "http://www.w3.org/2001/XMLSchema#dayTimeDuration". It SHALL return a result of  
 4337 "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration,  
 4338 then this function SHALL return the value by adding the corresponding negative duration, as per  
 4339 the specification [XS] Appendix E. If the second argument is a negative duration, then the result  
 4340 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-  
 4341 dayTimeDuration" had been applied to the corresponding positive duration.
- 4342 • urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-yearMonthDuration
- 4343 This function SHALL take two arguments, the first SHALL be a  
 4344 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a  
 4345 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of  
 4346 "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration,  
 4347 then this function SHALL return the value by adding the corresponding negative duration, as per  
 4348 the specification [XS] Appendix E. If the second argument is a negative duration, then the result  
 4349 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-  
 4350 yearMonthDuration" had been applied to the corresponding positive duration.
- 4351 • urn:oasis:names:tc:xacml:3.0:function:date-add-yearMonthDuration
- 4352 This function SHALL take two arguments, the first SHALL be a  
 4353 "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a  
 4354 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of  
 4355 "http://www.w3.org/2001/XMLSchema#date". This function SHALL return the value by adding the  
 4356 second argument to the first argument according to the specification of adding duration to date  
 4357 [XS] Appendix E.
- 4358 • urn:oasis:names:tc:xacml:3.0:function:date-subtract-yearMonthDuration
- 4359 This function SHALL take two arguments, the first SHALL be a  
 4360 "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a  
 4361 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of  
 4362 "http://www.w3.org/2001/XMLSchema#date". If the second argument is a positive duration, then  
 4363 this function SHALL return the value by adding the corresponding negative duration, as per the  
 4364 specification [XS] Appendix E. If the second argument is a negative duration, then the result  
 4365 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration"  
 4366 had been applied to the corresponding positive duration.

### 4367 A.3.8 Non-numeric comparison functions

4368 These functions perform comparison operations on two arguments of non-numerical types.

- 4369 • urn:oasis:names:tc:xacml:1.0:function:string-greater-than
- 4370 This function SHALL take two arguments of data-type  
 4371 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an  
 4372 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first  
 4373 argument is lexicographically strictly greater than the second argument. Otherwise, it SHALL  
 4374 return "False". The comparison SHALL use Unicode codepoint collation, as defined for the  
 4375 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [XF].
- 4376 • urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal
- 4377 This function SHALL take two arguments of data-type  
 4378 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an  
 4379 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first  
 4380 argument is lexicographically greater than or equal to the second argument. Otherwise, it SHALL  
 4381 return "False". The comparison SHALL use Unicode codepoint collation, as defined for the  
 4382 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [XF].



- 4383 • urn:oasis:names:tc:xacml:1.0:function:string-less-than
  - 4384 This function SHALL take two arguments of data-type
  - 4385 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
  - 4386 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only the first
  - 4387 argument is lexicographically strictly less than the second argument. Otherwise, it SHALL return
  - 4388 "False". The comparison SHALL use Unicode codepoint collation, as defined for the identifier
  - 4389 http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.
- 4390 • urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal
  - 4391 This function SHALL take two arguments of data-type
  - 4392 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
  - 4393 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only the first
  - 4394 argument is lexicographically less than or equal to the second argument. Otherwise, it SHALL
  - 4395 return "False". The comparison SHALL use Unicode codepoint collation, as defined for the
  - 4396 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.
- 4397 • urn:oasis:names:tc:xacml:1.0:function:time-greater-than
  - 4398 This function SHALL take two arguments of data-type
  - 4399 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
  - 4400 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
  - 4401 argument is greater than the second argument according to the order relation specified for
  - 4402 "http://www.w3.org/2001/XMLSchema#time" **[XS]** Section 3.2.8. Otherwise, it SHALL return
  - 4403 "False". Note: it is illegal to compare a time that includes a time-zone value with one that does
  - 4404 not. In such cases, the time-in-range function should be used.
- 4405 • urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal
  - 4406 This function SHALL take two arguments of data-type
  - 4407 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
  - 4408 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
  - 4409 argument is greater than or equal to the second argument according to the order relation
  - 4410 specified for "http://www.w3.org/2001/XMLSchema#time" **[XS]** Section 3.2.8. Otherwise, it
  - 4411 SHALL return "False". Note: it is illegal to compare a time that includes a time-zone value with
  - 4412 one that does not. In such cases, the time-in-range function should be used.
- 4413 • urn:oasis:names:tc:xacml:1.0:function:time-less-than
  - 4414 This function SHALL take two arguments of data-type
  - 4415 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
  - 4416 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
  - 4417 argument is less than the second argument according to the order relation specified for
  - 4418 "http://www.w3.org/2001/XMLSchema#time" **[XS]** Section 3.2.8. Otherwise, it SHALL return
  - 4419 "False". Note: it is illegal to compare a time that includes a time-zone value with one that does
  - 4420 not. In such cases, the time-in-range function should be used.
- 4421 • urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal
  - 4422 This function SHALL take two arguments of data-type
  - 4423 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
  - 4424 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
  - 4425 argument is less than or equal to the second argument according to the order relation specified
  - 4426 for "http://www.w3.org/2001/XMLSchema#time" **[XS]** Section 3.2.8. Otherwise, it SHALL return
  - 4427 "False". Note: it is illegal to compare a time that includes a time-zone value with one that does
  - 4428 not. In such cases, the time-in-range function should be used.
- 4429 • urn:oasis:names:tc:xacml:2.0:function:time-in-range
  - 4430 This function SHALL take three arguments of data-type
  - 4431 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
  - 4432 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the first argument falls
  - 4433 in the range defined inclusively by the second and third arguments. Otherwise, it SHALL return

4434 "False". Regardless of its value, the third argument SHALL be interpreted as a time that is equal  
4435 to, or later than by less than twenty-four hours, the second argument. If no time zone is provided  
4436 for the first argument, it SHALL use the default time zone at the **context handler**. If no time zone  
4437 is provided for the second or third arguments, then they SHALL use the time zone from the first  
4438 argument.

4439 • urn:oasis:names:tc:xacml:1.0:function:date-time-greater-than

4440 This function SHALL take two arguments of data-type  
4441 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an  
4442 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first  
4443 argument is greater than the second argument according to the order relation specified for  
4444 "http://www.w3.org/2001/XMLSchema#dateTime" by [XS] part 2, section 3.2.7. Otherwise, it  
4445 SHALL return "False". Note: if a dateTime value does not include a time-zone value, then an  
4446 implicit time-zone value SHALL be assigned, as described in [XS].

4447 • urn:oasis:names:tc:xacml:1.0:function:date-time-greater-than-or-equal

4448 This function SHALL take two arguments of data-type  
4449 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an  
4450 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first  
4451 argument is greater than or equal to the second argument according to the order relation  
4452 specified for "http://www.w3.org/2001/XMLSchema#dateTime" by [XS] part 2, section 3.2.7.  
4453 Otherwise, it SHALL return "False". Note: if a dateTime value does not include a time-zone  
4454 value, then an implicit time-zone value SHALL be assigned, as described in [XS].

4455 • urn:oasis:names:tc:xacml:1.0:function:date-time-less-than

4456 This function SHALL take two arguments of data-type  
4457 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an  
4458 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first  
4459 argument is less than the second argument according to the order relation specified for  
4460 "http://www.w3.org/2001/XMLSchema#dateTime" by [XS, part 2, section 3.2.7]. Otherwise, it  
4461 SHALL return "False". Note: if a dateTime value does not include a time-zone value, then an  
4462 implicit time-zone value SHALL be assigned, as described in [XS].

4463 • urn:oasis:names:tc:xacml:1.0:function:date-time-less-than-or-equal

4464 This function SHALL take two arguments of data-type "http://www.w3.org/2001/XMLSchema#  
4465 dateTime" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL  
4466 return "True" if and only if the first argument is less than or equal to the second argument  
4467 according to the order relation specified for "http://www.w3.org/2001/XMLSchema#dateTime" by  
4468 [XS] part 2, section 3.2.7. Otherwise, it SHALL return "False". Note: if a dateTime value does  
4469 not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described  
4470 in [XS].

4471 • urn:oasis:names:tc:xacml:1.0:function:date-greater-than

4472 This function SHALL take two arguments of data-type  
4473 "http://www.w3.org/2001/XMLSchema#date" and SHALL return an  
4474 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first  
4475 argument is greater than the second argument according to the order relation specified for  
4476 "http://www.w3.org/2001/XMLSchema#date" by [XS] part 2, section 3.2.9. Otherwise, it SHALL  
4477 return "False". Note: if a date value does not include a time-zone value, then an implicit time-  
4478 zone value SHALL be assigned, as described in [XS].

4479 • urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal

4480 This function SHALL take two arguments of data-type  
4481 "http://www.w3.org/2001/XMLSchema#date" and SHALL return an  
4482 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first  
4483 argument is greater than or equal to the second argument according to the order relation  
4484 specified for "http://www.w3.org/2001/XMLSchema#date" by [XS] part 2, section 3.2.9.

4485           Otherwise, it SHALL return "False". Note: if a date value does not include a time-zone value,  
4486           then an implicit time-zone value SHALL be assigned, as described in [XS].

4487   •   urn:oasis:names:tc:xacml:1.0:function:date-less-than

4488           This function SHALL take two arguments of data-type  
4489           "http://www.w3.org/2001/XMLSchema#date" and SHALL return an  
4490           "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first  
4491           argument is less than the second argument according to the order relation specified for  
4492           "http://www.w3.org/2001/XMLSchema#date" by [XS] part 2, section 3.2.9. Otherwise, it SHALL  
4493           return "False". Note: if a date value does not include a time-zone value, then an implicit time-  
4494           zone value SHALL be assigned, as described in [XS].

4495   •   urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal

4496           This function SHALL take two arguments of data-type  
4497           "http://www.w3.org/2001/XMLSchema#date" and SHALL return an  
4498           "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first  
4499           argument is less than or equal to the second argument according to the order relation specified  
4500           for "http://www.w3.org/2001/XMLSchema#date" by [XS] part 2, section 3.2.9. Otherwise, it  
4501           SHALL return "False". Note: if a date value does not include a time-zone value, then an implicit  
4502           time-zone value SHALL be assigned, as described in [XS].

### 4503   A.3.9 String functions

4504   The following functions operate on strings and convert to and from other data types.

4505   •   urn:oasis:names:tc:xacml:2.0:function:string-concatenate

4506           This function SHALL take two or more arguments of data-type  
4507           "http://www.w3.org/2001/XMLSchema#string" and SHALL return a  
4508           "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the concatenation, in order,  
4509           of the arguments.

4510   •   urn:oasis:names:tc:xacml:3.0:function:boolean-from-string

4511           This function SHALL take one argument of data-type  
4512           "http://www.w3.org/2001/XMLSchema#string", and SHALL return an  
4513           "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be the string converted to a  
4514           boolean. If the argument is not a valid lexical representation of a boolean, then the result SHALL  
4515           be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

4516   •   urn:oasis:names:tc:xacml:3.0:function:string-from-boolean

4517           This function SHALL take one argument of data-type  
4518           "http://www.w3.org/2001/XMLSchema#boolean", and SHALL return an  
4519           "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the boolean converted to a  
4520           string in the canonical form specified in [XS].

4521   •   urn:oasis:names:tc:xacml:3.0:function:integer-from-string

4522           This function SHALL take one argument of data-type  
4523           "http://www.w3.org/2001/XMLSchema#string", and SHALL return an  
4524           "http://www.w3.org/2001/XMLSchema#integer". The result SHALL be the string converted to an  
4525           integer. If the argument is not a valid lexical representation of an integer, then the result SHALL  
4526           be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

4527   •   urn:oasis:names:tc:xacml:3.0:function:string-from-integer

4528           This function SHALL take one argument of data-type  
4529           "http://www.w3.org/2001/XMLSchema#integer", and SHALL return an  
4530           "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the integer converted to a  
4531           string in the canonical form specified in [XS].

4532   •   urn:oasis:names:tc:xacml:3.0:function:double-from-string

4533 This function SHALL take one argument of data-type  
4534 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an  
4535 "http://www.w3.org/2001/XMLSchema#double". The result SHALL be the string converted to a  
4536 double. If the argument is not a valid lexical representation of a double, then the result SHALL be  
4537 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

4538 • urn:oasis:names:tc:xacml:3.0:function:string-from-double  
4539 This function SHALL take one argument of data-type  
4540 "http://www.w3.org/2001/XMLSchema#double", and SHALL return an  
4541 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the double converted to a  
4542 string in the canonical form specified in **[XS]**.

4543 • urn:oasis:names:tc:xacml:3.0:function:time-from-string  
4544 This function SHALL take one argument of data-type  
4545 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an  
4546 "http://www.w3.org/2001/XMLSchema#time". The result SHALL be the string converted to a time.  
4547 If the argument is not a valid lexical representation of a time, then the result SHALL be  
4548 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

4549 • urn:oasis:names:tc:xacml:3.0:function:string-from-time  
4550 This function SHALL take one argument of data-type  
4551 "http://www.w3.org/2001/XMLSchema#time", and SHALL return an  
4552 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the time converted to a  
4553 string in the canonical form specified in **[XS]**.

4554 • urn:oasis:names:tc:xacml:3.0:function:date-from-string  
4555 This function SHALL take one argument of data-type  
4556 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an  
4557 "http://www.w3.org/2001/XMLSchema#date". The result SHALL be the string converted to a  
4558 date. If the argument is not a valid lexical representation of a date, then the result SHALL be  
4559 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

4560 • urn:oasis:names:tc:xacml:3.0:function:string-from-date  
4561 This function SHALL take one argument of data-type  
4562 "http://www.w3.org/2001/XMLSchema#date", and SHALL return an  
4563 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the date converted to a  
4564 string in the canonical form specified in **[XS]**.

4565 • urn:oasis:names:tc:xacml:3.0:function:dateTime-from-string  
4566 This function SHALL take one argument of data-type  
4567 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an  
4568 "http://www.w3.org/2001/XMLSchema#dateTime". The result SHALL be the string converted to a  
4569 dateTime. If the argument is not a valid lexical representation of a dateTime, then the result  
4570 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

4571 urn:oasis:names:tc:xacml:3.0:function:string-from-dateTime  
4572 This function SHALL take one argument of data-type  
4573 "http://www.w3.org/2001/XMLSchema#dateTime", and SHALL return an  
4574 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the dateTime converted to a  
4575 string in the canonical form specified in **[XS]**.

4576 • urn:oasis:names:tc:xacml:3.0:function:anyURI-from-string  
4577 This function SHALL take one argument of data-type  
4578 "http://www.w3.org/2001/XMLSchema#string", and SHALL return a  
4579 "http://www.w3.org/2001/XMLSchema#anyURI". The result SHALL be the URI constructed by  
4580 converting the argument to an URI. If the argument is not a valid lexical representation of a URI,  
4581 then the result SHALL be Indeterminate with status code  
4582 urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- 4583 • urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI
  - 4584 This function SHALL take one argument of data-type
  - 4585 "http://www.w3.org/2001/XMLSchema#anyURI", and SHALL return an
  - 4586 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the URI converted to a
  - 4587 string in the form it was originally represented in XML form.
- 4588 • urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-from-string
  - 4589 This function SHALL take one argument of data-type
  - 4590 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
  - 4591 "http://www.w3.org/2001/XMLSchema#dayTimeDuration ". The result SHALL be the string
  - 4592 converted to a dayTimeDuration. If the argument is not a valid lexical representation of a
  - 4593 dayTimeDuration, then the result SHALL be Indeterminate with status code
  - 4594 urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4595 • urn:oasis:names:tc:xacml:3.0:function:string-from-dayTimeDuration
  - 4596 This function SHALL take one argument of data-type
  - 4597 "http://www.w3.org/2001/XMLSchema#dayTimeDuration ", and SHALL return an
  - 4598 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the dayTimeDuration
  - 4599 converted to a string in the canonical form specified in **[XPathFunc]**.
- 4600 • urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-from-string
  - 4601 This function SHALL take one argument of data-type
  - 4602 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
  - 4603 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". The result SHALL be the string
  - 4604 converted to a yearMonthDuration. If the argument is not a valid lexical representation of a
  - 4605 yearMonthDuration, then the result SHALL be Indeterminate with status code
  - 4606 urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4607 • urn:oasis:names:tc:xacml:3.0:function:string-from-yearMonthDuration
  - 4608 This function SHALL take one argument of data-type
  - 4609 "http://www.w3.org/2001/XMLSchema#yearMonthDuration", and SHALL return an
  - 4610 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the yearMonthDuration
  - 4611 converted to a string in the canonical form specified in **[XPathFunc]**.
- 4612 • urn:oasis:names:tc:xacml:3.0:function:x500Name-from-string
  - 4613 This function SHALL take one argument of data-type
  - 4614 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
  - 4615 "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". The result SHALL be the string converted
  - 4616 to an x500Name. If the argument is not a valid lexical representation of a X500Name, then the
  - 4617 result SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4618 • urn:oasis:names:tc:xacml:3.0:function:string-from-x500Name
  - 4619 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:1.0:data-
  - 4620 type:x500Name", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result
  - 4621 SHALL be the x500Name converted to a string in the form it was originally represented in XML
  - 4622 form..
- 4623 • urn:oasis:names:tc:xacml:3.0:function:rfc822Name-from-string
  - 4624 This function SHALL take one argument of data-type
  - 4625 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
  - 4626 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". The result SHALL be the string converted
  - 4627 to an rfc822Name. If the argument is not a valid lexical representation of an rfc822Name, then the
  - 4628 result SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4629 • urn:oasis:names:tc:xacml:3.0:function:string-from-rfc822Name
  - 4630 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:1.0:data-
  - 4631 type:rfc822Name", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The

- 4632 result SHALL be the rfc822Name converted to a string in the form it was originally represented in  
4633 XML form.
- 4634 • urn:oasis:names:tc:xacml:3.0:function:ipAddress-from-string  
4635 This function SHALL take one argument of data-type  
4636 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an  
4637 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". The result SHALL be the string converted to  
4638 an ipAddress. If the argument is not a valid lexical representation of an ipAddress, then the result  
4639 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4640 • urn:oasis:names:tc:xacml:3.0:function:string-from-ipAddress  
4641 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:2.0:data-  
4642 type:ipAddress", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result  
4643 SHALL be the ipAddress converted to a string in the form it was originally represented in XML  
4644 form.
- 4645 • urn:oasis:names:tc:xacml:3.0:function:dnsName-from-string  
4646 This function SHALL take one argument of data-type  
4647 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an  
4648 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". The result SHALL be the string converted to  
4649 a dnsName. If the argument is not a valid lexical representation of a dnsName, then the result  
4650 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4651 • urn:oasis:names:tc:xacml:3.0:function:string-from-dnsName  
4652 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:2.0:data-  
4653 type:dnsName", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result  
4654 SHALL be the dnsName converted to a string in the form it was originally represented in XML  
4655 form.
- 4656 • urn:oasis:names:tc:xacml:3.0:function:string-starts-with  
4657 This function SHALL take two arguments of data-type  
4658 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a  
4659 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string  
4660 begins with the first string, and false otherwise. Equality testing SHALL be done as defined for  
4661 urn:oasis:names:tc:xacml:1.0:function:string-equal.
- 4662 • urn:oasis:names:tc:xacml:3.0:function:anyURI-starts-with  
4663 This function SHALL take a first argument of data-  
4664 type"http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type  
4665 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a  
4666 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted  
4667 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI begins with the string,  
4668 and false otherwise. Equality testing SHALL be done as defined for  
4669 urn:oasis:names:tc:xacml:1.0:function:string-equal.
- 4670 • urn:oasis:names:tc:xacml:3.0:function:string-ends-with  
4671 This function SHALL take two arguments of data-type  
4672 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a  
4673 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string  
4674 ends with the first string, and false otherwise. Equality testing SHALL be done as defined for  
4675 urn:oasis:names:tc:xacml:1.0:function:string-equal.
- 4676 • urn:oasis:names:tc:xacml:3.0:function:anyURI-ends-with  
4677 This function SHALL take a first argument of data-type  
4678 "http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type  
4679 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a  
4680 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted  
4681 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI ends with the string,

4682 and false otherwise. Equality testing SHALL be done as defined for  
4683 urn:oasis:names:tc:xacml:1.0:function:string-equal.

4684 • urn:oasis:names:tc:xacml:3.0:function:string-contains  
4685 This function SHALL take two arguments of data-type  
4686 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a  
4687 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string  
4688 contains the first string, and false otherwise. Equality testing SHALL be done as defined for  
4689 urn:oasis:names:tc:xacml:1.0:function:string-equal.

4690 • urn:oasis:names:tc:xacml:3.0:function:anyURI-contains  
4691 This function SHALL take a first argument of data-type  
4692 "http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type  
4693 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a  
4694 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted  
4695 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI contains the string, and  
4696 false otherwise. Equality testing SHALL be done as defined for  
4697 urn:oasis:names:tc:xacml:1.0:function:string-equal.

4698 • urn:oasis:names:tc:xacml:3.0:function:string-substring  
4699 This function SHALL take a first argument of data-type  
4700 "http://www.w3.org/2001/XMLSchema#string" and a second and a third argument of type  
4701 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return a  
4702 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the substring of the first  
4703 argument beginning at the position given by the second argument and ending at the position  
4704 before the position given by the third argument. The first character of the string has position zero.  
4705 The negative integer value -1 given for the third arguments indicates the end of the string. If the  
4706 second or third arguments are out of bounds, then the function MUST evaluate to Indeterminate  
4707 with a status code of urn:oasis:names:tc:xacml:1.0:status:processing-error.

4708 • urn:oasis:names:tc:xacml:3.0:function:anyURI-substring  
4709 This function SHALL take a first argument of data-type  
4710 "http://www.w3.org/2001/XMLSchema#anyURI" and a second and a third argument of type  
4711 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return a  
4712 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the substring of the first  
4713 argument converted to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI  
4714 beginning at the position given by the second argument and ending at the position before the  
4715 position given by the third argument. The first character of the URI converted to a string has  
4716 position zero. The negative integer value -1 given for the third arguments indicates the end of the  
4717 string. If the second or third arguments are out of bounds, then the function MUST evaluate to  
4718 Indeterminate with a status code of  
4719 urn:oasis:names:tc:xacml:1.0:status:processing-error. If the resulting substring  
4720 is not syntactically a valid URI, then the function MUST evaluate to Indeterminate with a status  
4721 code of urn:oasis:names:tc:xacml:1.0:status:processing-error.

4722

### 4723 A.3.10 Bag functions

4724 These functions operate on a **bag** of 'type' values, where type is one of the primitive data-types, and x.x  
4725 is a version of XACML where the function has been defined. Some additional conditions defined for  
4726 each function below SHALL cause the expression to evaluate to "Indeterminate".

4727 • urn:oasis:names:tc:xacml:x.x:function:type-one-and-only  
4728 This function SHALL take a **bag** of 'type' values as an argument and SHALL return a value of  
4729 'type'. It SHALL return the only value in the **bag**. If the **bag** does not have one and only one  
4730 value, then the expression SHALL evaluate to "Indeterminate".

- 4731 • urn:oasis:names:tc:xacml:x.x:function:type-bag-size
- 4732 This function SHALL take a **bag** of 'type' values as an argument and SHALL return an
- 4733 "http://www.w3.org/2001/XMLSchema#integer" indicating the number of values in the **bag**.
- 4734 • urn:oasis:names:tc:xacml:x.x:function:type-is-in
- 4735 This function SHALL take an argument of 'type' as the first argument and a **bag** of 'type' values
- 4736 as the second argument and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean".
- 4737 The function SHALL evaluate to "True" if and only if the first argument matches by the
- 4738 "urn:oasis:names:tc:xacml:x.x:function:type-equal" any value in the **bag**. Otherwise, it SHALL
- 4739 return "False".
- 4740 • urn:oasis:names:tc:xacml:x.x:function:type-bag
- 4741 This function SHALL take any number of arguments of 'type' and return a **bag** of 'type' values
- 4742 containing the values of the arguments. An application of this function to zero arguments SHALL
- 4743 produce an empty **bag** of the specified data-type.

### 4744 A.3.11 Set functions

4745 These functions operate on **bags** mimicking sets by eliminating duplicate elements from a **bag**.

- 4746 • urn:oasis:names:tc:xacml:x.x:function:type-intersection
- 4747 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4748 **bag** of 'type' values such that it contains only elements that are common between the two **bags**,
- 4749 which is determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal". No duplicates, as
- 4750 determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal", SHALL exist in the result.
- 4751 • urn:oasis:names:tc:xacml:x.x:function:type-at-least-one-member-of
- 4752 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4753 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL evaluate to "True" if and
- 4754 only if at least one element of the first argument is contained in the second argument as
- 4755 determined by "urn:oasis:names:tc:xacml:x.x:function:type-is-in".
- 4756 • urn:oasis:names:tc:xacml:x.x:function:type-union
- 4757 This function SHALL take two or more arguments that are both a **bag** of 'type' values. The
- 4758 expression SHALL return a **bag** of 'type' such that it contains all elements of all the argument
- 4759 **bags**. No duplicates, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal",
- 4760 SHALL exist in the result.
- 4761 • urn:oasis:names:tc:xacml:x.x:function:type-subset
- 4762 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4763 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
- 4764 argument is a subset of the second argument. Each argument SHALL be considered to have had
- 4765 its duplicates removed, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal",
- 4766 before the subset calculation.
- 4767 • urn:oasis:names:tc:xacml:x.x:function:type-set-equals
- 4768 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4769 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return the result of applying
- 4770 "urn:oasis:names:tc:xacml:1.0:function:and" to the application of
- 4771 "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the first and second arguments and the
- 4772 application of "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the second and first
- 4773 arguments.

### 4774 A.3.12 Higher-order bag functions

4775 This section describes functions in XACML that perform operations on **bags** such that functions may be

4776 applied to the **bags** in general.



4777 • urn:oasis:names:tc:xacml:3.0:function:any-of

4778 This function applies a Boolean function between specific primitive values and a **bag** of values,  
4779 and SHALL return "True" if and only if the **predicate** is "True" for at least one element of the **bag**.

4780 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL  
4781 be an <Function> element that names a Boolean function that takes n arguments of primitive  
4782 types. Under the remaining n arguments, n-1 parameters SHALL be values of primitive data-  
4783 types and one SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated as  
4784 if the function named in the <Function> argument were applied to the n-1 non-bag arguments  
4785 and each element of the bag argument and the results are combined with  
4786 "urn:oasis:names:tc:xacml:1.0:function:or".

4787 For example, the following expression SHALL return "True":

```
4788 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">  
4789   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>  
4790   <AttributeValue  
4791     DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4792   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4793     <AttributeValue  
4794       DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>  
4795     <AttributeValue  
4796       DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4797     <AttributeValue  
4798       DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>  
4799     <AttributeValue  
4800       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4801   </Apply>  
4802 </Apply>
```

4803 This expression is "True" because the first argument is equal to at least one of the elements of  
4804 the **bag**, according to the function.

4805 • urn:oasis:names:tc:xacml:3.0:function:all-of

4806 This function applies a Boolean function between a specific primitive value and a **bag** of values,  
4807 and returns "True" if and only if the **predicate** is "True" for every element of the **bag**.

4808 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL  
4809 be a <Function> element that names a Boolean function that takes n arguments of primitive  
4810 types. Under the remaining n arguments, n-1 parameters SHALL be values of primitive data-  
4811 types and one SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated as  
4812 if the function named in the <Function> argument were applied to the n-1 non-bag arguments  
4813 and each element of the bag argument and the results are combined with  
4814 "urn:oasis:names:tc:xacml:1.0:function:and".

4815 For example, the following expression SHALL evaluate to "True":

```
4816 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:all-of">  
4817   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4818     greater-than"/>  
4819   <AttributeValue  
4820     DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>  
4821   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4822     <AttributeValue  
4823       DataType="http://www.w3.org/2001/XMLSchema#integer">9</AttributeValue>  
4824     <AttributeValue  
4825       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>  
4826     <AttributeValue  
4827       DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>  
4828     <AttributeValue  
4829       DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>  
4830   </Apply>  
4831 </Apply>
```

4832 This expression is "True" because the first argument (10) is greater than all of the elements of the  
4833 **bag** (9,3,4 and 2).

4834 • urn:oasis:names:tc:xacml:3.0:function:any-of-any

4835 This function applies a Boolean function on each tuple from the cross product on all bags  
4836 arguments, and returns "True" if and only if the **predicate** is "True" for at least one inside-function  
4837 call.

4838 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL  
4839 be an <Function> element that names a Boolean function that takes n arguments. The  
4840 remaining arguments are either primitive data types or bags of primitive types. The expression  
4841 SHALL be evaluated as if the function named in the <Function> argument was applied between  
4842 every tuple of the cross product on all bags and the primitive values, and the results were  
4843 combined using "urn:oasis:names:tc:xacml:1.0:function:or". The semantics are that the result of  
4844 the expression SHALL be "True" if and only if the applied **predicate** is "True" for at least one  
4845 function call on the tuples from the **bags** and primitive values.

4846 For example, the following expression SHALL evaluate to "True":

```
4847 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of-any">  
4848 <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>  
4849 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4850 <AttributeValue  
4851 DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4852 <AttributeValue  
4853 DataType="http://www.w3.org/2001/XMLSchema#string">Mary</AttributeValue>  
4854 </Apply>  
4855 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4856 <AttributeValue  
4857 DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>  
4858 <AttributeValue  
4859 DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4860 <AttributeValue  
4861 DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>  
4862 <AttributeValue  
4863 DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4864 </Apply>  
4865 </Apply>
```

4866 This expression is "True" because at least one of the elements of the first **bag**, namely "Ringo", is  
4867 equal to at least one of the elements of the second **bag**.

4868 • urn:oasis:names:tc:xacml:1.0:function:all-of-any

4869 This function applies a Boolean function between the elements of two **bags**. The expression  
4870 SHALL be "True" if and only if the supplied **predicate** is "True" between each element of the first  
4871 **bag** and any element of the second **bag**.

4872 This function SHALL take three arguments. The first argument SHALL be an <Function>  
4873 element that names a Boolean function that takes two arguments of primitive types. The second  
4874 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a  
4875 primitive data-type. The expression SHALL be evaluated as if the  
4876 "urn:oasis:names:tc:xacml:3.0:function:any-of" function had been applied to each value of the first  
4877 **bag** and the whole of the second **bag** using the supplied xacml:Function, and the results were  
4878 then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

4879 For example, the following expression SHALL evaluate to "True":

```
4880 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-any">  
4881 <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4882 greater-than"/>  
4883 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4884 <AttributeValue  
4885 DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
```

```

4886         <AttributeValue
4887         DataType="http://www.w3.org/2001/XMLSchema#integer">20</AttributeValue>
4888         </Apply>
4889         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4890         <AttributeValue
4891         DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4892         <AttributeValue
4893         DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4894         <AttributeValue
4895         DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4896         <AttributeValue
4897         DataType="http://www.w3.org/2001/XMLSchema#integer">19</AttributeValue>
4898         </Apply>
4899     </Apply>

```

4900 This expression is "True" because each of the elements of the first **bag** is greater than at least  
4901 one of the elements of the second **bag**.

4902 • urn:oasis:names:tc:xacml:1.0:function:any-of-all

4903 This function applies a Boolean function between the elements of two **bags**. The expression  
4904 SHALL be "True" if and only if the supplied **predicate** is "True" between each element of the  
4905 second **bag** and any element of the first **bag**.

4906 This function SHALL take three arguments. The first argument SHALL be an <Function>  
4907 element that names a Boolean function that takes two arguments of primitive types. The second  
4908 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a  
4909 primitive data-type. The expression SHALL be evaluated as if the  
4910 "urn:oasis:names:tc:xacml:3.0:function:any-of" function had been applied to each value of the  
4911 second **bag** and the whole of the first **bag** using the supplied xacml:Function, and the results  
4912 were then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

4913 For example, the following expression SHALL evaluate to "True":

```

4914 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-all">
4915 <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
4916 greater-than"/>
4917 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4918 <AttributeValue
4919 DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4920 <AttributeValue
4921 DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4922 </Apply>
4923 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4924 <AttributeValue
4925 DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4926 <AttributeValue
4927 DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
4928 <AttributeValue
4929 DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4930 <AttributeValue
4931 DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
4932 </Apply>
4933 </Apply>

```

4934 This expression is "True" because, for all of the values in the second **bag**, there is a value in the  
4935 first **bag** that is greater.

4936 • urn:oasis:names:tc:xacml:1.0:function:all-of-all

4937 This function applies a Boolean function between the elements of two **bags**. The expression  
4938 SHALL be "True" if and only if the supplied **predicate** is "True" between each and every element  
4939 of the first **bag** collectively against all the elements of the second **bag**.

4940 This function SHALL take three arguments. The first argument SHALL be an <Function>  
4941 element that names a Boolean function that takes two arguments of primitive types. The second

4942 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a  
4943 primitive data-type. The expression is evaluated as if the function named in the <Function>  
4944 element were applied between every element of the second argument and every element of the  
4945 third argument and the results were combined using "urn:oasis:names:tc:xacml:1.0:function:and".  
4946 The semantics are that the result of the expression is "True" if and only if the applied **predicate** is  
4947 "True" for all elements of the first **bag** compared to all the elements of the second **bag**.

4948 For example, the following expression SHALL evaluate to "True":

```
4949 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-all">  
4950 <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4951 greater-than"/>  
4952 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4953 <AttributeValue  
4954 DataType="http://www.w3.org/2001/XMLSchema#integer">6</AttributeValue>  
4955 <AttributeValue  
4956 DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>  
4957 </Apply>  
4958 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4959 <AttributeValue  
4960 DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>  
4961 <AttributeValue  
4962 DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>  
4963 <AttributeValue  
4964 DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>  
4965 <AttributeValue  
4966 DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>  
4967 </Apply>  
4968 </Apply>
```

4969 This expression is "True" because all elements of the first **bag**, "5" and "6", are each greater than  
4970 all of the integer values "1", "2", "3", "4" of the second **bag**.

4971 • urn:oasis:names:tc:xacml:3.0:function:map

4972 This function converts a **bag** of values to another **bag** of values.

4973 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL  
4974 be a <Function> element naming a function that takes a n arguments of a primitive data-type  
4975 and returns a value of a primitive data-type Under the remaining n arguments, n-1 parameters  
4976 SHALL be values of primitive data-types and one SHALL be a **bag** of a primitive data-type. The  
4977 expression SHALL be evaluated as if the function named in the <Function> argument were  
4978 applied to the n-1 non-bag arguments and each element of the bag argument and resulting in a  
4979 **bag** of the converted value. The result SHALL be a **bag** of the primitive data-type that is returned  
4980 by the function named in the <xacml:Function> element.

4981 For example, the following expression,

```
4982 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:map">  
4983 <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-  
4984 normalize-to-lower-case">  
4985 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4986 <AttributeValue  
4987 DataType="http://www.w3.org/2001/XMLSchema#string">Hello</AttributeValue>  
4988 <AttributeValue  
4989 DataType="http://www.w3.org/2001/XMLSchema#string">World!</AttributeValue>  
4990 </Apply>  
4991 </Apply>
```

4992 evaluates to a **bag** containing "hello" and "world!".

### 4993 A.3.13 Regular-expression-based functions

4994 These functions operate on various types using regular expressions and evaluate to  
4995 "http://www.w3.org/2001/XMLSchema#boolean".

- 4996 • urn:oasis:names:tc:xacml:1.0:function:string-regexp-match
- 4997 This function decides a regular expression match. It SHALL take two arguments of  
 4998 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an  
 4999 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular  
 5000 expression and the second argument SHALL be a general string. The function specification  
 5001 SHALL be that of the "xf:matches" function with the arguments reversed [XF] Section 7.6.2.
- 5002 • urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match
- 5003 This function decides a regular expression match. It SHALL take two arguments; the first is of  
 5004 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type  
 5005 "http://www.w3.org/2001/XMLSchema#anyURI". It SHALL return an  
 5006 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular  
 5007 expression and the second argument SHALL be a URI. The function SHALL convert the second  
 5008 argument to type "http://www.w3.org/2001/XMLSchema#string" with  
 5009 urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI, then apply  
 5010 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5011 • urn:oasis:names:tc:xacml:2.0:function:ipAddress-regexp-match
- 5012 This function decides a regular expression match. It SHALL take two arguments; the first is of  
 5013 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type  
 5014 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". It SHALL return an  
 5015 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular  
 5016 expression and the second argument SHALL be an IPv4 or IPv6 address. The function SHALL  
 5017 convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with  
 5018 urn:oasis:names:tc:xacml:3.0:function:string-from-ipAddress, then apply  
 5019 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5020 • urn:oasis:names:tc:xacml:2.0:function:dnsName-regexp-match
- 5021 This function decides a regular expression match. It SHALL take two arguments; the first is of  
 5022 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type  
 5023 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". It SHALL return an  
 5024 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular  
 5025 expression and the second argument SHALL be a DNS name. The function SHALL convert the  
 5026 second argument to type "http://www.w3.org/2001/XMLSchema#string" with  
 5027 urn:oasis:names:tc:xacml:3.0:function:string-from-dnsName, then apply  
 5028 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5029 • urn:oasis:names:tc:xacml:2.0:function:rfc822Name-regexp-match
- 5030 This function decides a regular expression match. It SHALL take two arguments; the first is of  
 5031 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type  
 5032 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". It SHALL return an  
 5033 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular  
 5034 expression and the second argument SHALL be an RFC 822 name. The function SHALL convert  
 5035 the second argument to type "http://www.w3.org/2001/XMLSchema#string" with  
 5036 urn:oasis:names:tc:xacml:3.0:function:string-from-rfc822Name, then apply  
 5037 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5038 • urn:oasis:names:tc:xacml:2.0:function:x500Name-regexp-match
- 5039 This function decides a regular expression match. It SHALL take two arguments; the first is of  
 5040 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type  
 5041 "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". It SHALL return an  
 5042 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular  
 5043 expression and the second argument SHALL be an X.500 directory name. The function SHALL  
 5044 convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with  
 5045 urn:oasis:names:tc:xacml:3.0:function:string-from-x500Name, then apply  
 5046 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

### 5047 **A.3.14 Special match functions**

5048 These functions operate on various types and evaluate to  
5049 "http://www.w3.org/2001/XMLSchema#boolean" based on the specified standard matching algorithm.

- 5050 • urn:oasis:names:tc:xacml:1.0:function:x500Name-match

5051 This function shall take two arguments of "urn:oasis:names:tc:xacml:1.0:data-type:x500Name"  
5052 and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and  
5053 only if the first argument matches some terminal sequence of RDNs from the second argument  
5054 when compared using x500Name-equal.

5055 As an example (non-normative), if the first argument is "O=Medico Corp,C=US" and the second  
5056 argument is "cn=John Smith,o=Medico Corp,c=US", then the function will return "True".

- 5057 • urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match

5058 This function SHALL take two arguments, the first is of data-type  
5059 "http://www.w3.org/2001/XMLSchema#string" and the second is of data-type  
5060 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" and SHALL return an  
5061 "http://www.w3.org/2001/XMLSchema#boolean". This function SHALL evaluate to "True" if the  
5062 first argument matches the second argument according to the following specification.

5063 An RFC822 name consists of a local-part followed by "@" followed by a domain-part. The local-  
5064 part is case-sensitive, while the domain-part (which is usually a DNS name) is not case-sensitive.

5065 The second argument contains a complete rfc822Name. The first argument is a complete or  
5066 partial rfc822Name used to select appropriate values in the second argument as follows.

5067 In order to match a particular address in the second argument, the first argument must specify the  
5068 complete mail address to be matched. For example, if the first argument is  
5069 "Anderson@sun.com", this matches a value in the second argument of "Anderson@sun.com"  
5070 and "Anderson@SUN.COM", but not "Anne.Anderson@sun.com", "anderson@sun.com" or  
5071 "Anderson@east.sun.com".

5072 In order to match any address at a particular domain in the second argument, the first argument  
5073 must specify only a domain name (usually a DNS name). For example, if the first argument is  
5074 "sun.com", this matches a value in the second argument of "Anderson@sun.com" or  
5075 "Baxter@SUN.COM", but not "Anderson@east.sun.com".

5076 In order to match any address in a particular domain in the second argument, the first argument  
5077 must specify the desired domain-part with a leading ".". For example, if the first argument is  
5078 ".east.sun.com", this matches a value in the second argument of "Anderson@east.sun.com" and  
5079 "anne.anderson@ISRG.EAST.SUN.COM" but not "Anderson@sun.com".

### 5080 **A.3.15 XPath-based functions**

5081 This section specifies functions that take XPath expressions for arguments. An XPath expression  
5082 evaluates to a node-set, which is a set of XML nodes that match the expression. A node or node-set is  
5083 not in the formal data-type system of XACML. All comparison or other operations on node-sets are  
5084 performed in isolation of the particular function specified. The context nodes and namespace mappings  
5085 of the XPath expressions are defined by the XPath data-type, see section B.3. The following functions  
5086 are defined:

- 5087 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-count

5088 This function SHALL take an "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" as an  
5089 argument and evaluates to an "http://www.w3.org/2001/XMLSchema#integer". The value  
5090 returned from the function SHALL be the count of the nodes within the node-set that match the  
5091 given XPath expression. If the <Content> element of the category to which the XPath  
5092 expression applies to is not present in the request, this function SHALL return a value of zero.

- 5093 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-equal

5094 This function SHALL take two "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"  
5095 arguments and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". The function  
5096 SHALL return "True" if any of the XML nodes in the node-set matched by the first argument  
5097 equals any of the XML nodes in the node-set matched by the second argument. Two nodes are  
5098 considered equal if they have the same identity. If the <Content> element of the category to  
5099 which either XPath expression applies to is not present in the request, this function SHALL return  
5100 a value of "False".

5101 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-match

5102 This function SHALL take two "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"  
5103 arguments and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". This function  
5104 SHALL evaluate to "True" if one of the following two conditions is satisfied: (1) Any of the XML  
5105 nodes in the node-set matched by the first argument is equal to any of the XML nodes in the  
5106 node-set matched by the second argument; (2) any node below any of the XML nodes in the  
5107 node-set matched by the first argument is equal to any of the XML nodes in the node-set  
5108 matched by the second argument. Two nodes are considered equal if they have the same  
5109 identity. If the <Content> element of the category to which either XPath expression applies to is  
5110 not present in the request, this function SHALL return a value of "False".

5111 NOTE: The first **condition** is equivalent to "xpath-node-equal", and guarantees that "xpath-node-equal" is  
5112 a special case of "xpath-node-match".

### 5113 A.3.16 Other functions

5114 • urn:oasis:names:tc:xacml:3.0:function:access-permitted

5115 This function SHALL take an "http://www.w3.org/2001/XMLSchema#anyURI" and an  
5116 "http://www.w3.org/2001/XMLSchema#string" as arguments. The first argument SHALL be  
5117 interpreted as an **attribute** category. The second argument SHALL be interpreted as the XML  
5118 content of an <Attributes> element with **Category** equal to the first argument. The function  
5119 evaluates to an "http://www.w3.org/2001/XMLSchema#boolean". This function SHALL return  
5120 "True" if and only if the **policy** evaluation described below returns the value of "Permit".

5121 The following evaluation is described as if the **context** is actually instantiated, but it is only  
5122 required that an equivalent result be obtained.

5123 The function SHALL construct a new **context**, by copying all the information from the current  
5124 **context**, omitting any <Attributes> element with **Category** equal to the first argument. The  
5125 second function argument SHALL be added to the **context** as the content of an <Attributes>  
5126 element with **Category** equal to the first argument.

5127 The function SHALL invoke a complete **policy** evaluation using the newly constructed **context**.  
5128 This evaluation SHALL be completely isolated from the evaluation which invoked the function, but  
5129 shall use all current **policies** and combining algorithms, including any per request **policies**.

5130 The **PDP** SHALL detect any loop which may occur if successive evaluations invoke this function  
5131 by counting the number of total invocations of any instance of this function during any single initial  
5132 invocation of the **PDP**. If the total number of invocations exceeds the bound for such invocations,  
5133 the initial invocation of this function evaluates to Indeterminate with a  
5134 "urn:oasis:names:tc:xacml:1.0:status:processing-error" status code. Also, see the security  
5135 considerations in section 9.1.8.

### 5136 A.3.17 Extension functions and primitive types

5137 Functions and primitive types are specified by string identifiers allowing for the introduction of functions in  
5138 addition to those specified by XACML. This approach allows one to extend the XACML module with  
5139 special functions and special primitive data-types.

5140 In order to preserve the integrity of the XACML evaluation strategy, the result of an extension function  
5141 SHALL depend only on the values of its arguments. Global and hidden parameters SHALL NOT affect

5142 the evaluation of an expression. Functions SHALL NOT have side effects, as evaluation order cannot be  
5143 guaranteed in a standard way.

#### 5144 **A.4 Functions, data types, attributes and algorithms planned for** 5145 **deprecation**

5146 The following functions, data types and algorithms have been defined by previous versions of XACML  
5147 and newer and better alternatives are defined in XACML 3.0. Their use is discouraged for new use and  
5148 they are candidates for deprecation in future versions of XACML.

5149 The following xpath based functions have been replaced with equivalent functions which use the new  
5150 urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression datatype instead of strings.

- 5151 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-count
- 5152 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-count
- 5153 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal
- 5154 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-equal
- 5155 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-match
- 5156 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-match

5157 The following URI and string concatenation function has been replaced with a string to URI conversion  
5158 function, which allows the use of the general string functions with URI through string conversion.

- 5159 • urn:oasis:names:tc:xacml:2.0:function:uri-string-concatenate
- 5160 • Replaced by urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI

5161 The following identifiers have been replaced with official identifiers defined by W3C.

- 5162 • <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration>
- 5163 • Replaced with <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- 5164 • <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration>
- 5165 • Replaced with <http://www.w3.org/2001/XMLSchema#yearMonthDuration>

5166 The following functions have been replaced with functions which use the updated dayTimeDuration and  
5167 yearMonthDuration data types.

- 5168 • urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal
- 5169 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-equal
- 5170 • urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-equal
- 5171 • Replaced with urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-equal
- 5172 • urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration
- 5173 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-add-dayTimeDuration
- 5174 • urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration
- 5175 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-add-yearMonthDuration
- 5176 • urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration
- 5177 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-dayTimeDuration
- 5178 • urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-yearMonthDuration
- 5179 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-yearMonthDuration
- 5180 • urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration
- 5181 • Replaced with urn:oasis:names:tc:xacml:3.0:function:date-add-yearMonthDuration
- 5182 • urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration
- 5183 • Replaced with urn:oasis:names:tc:xacml:3.0:function:date-subtract-yearMonthDuration



- 5184 The following attribute identifiers have been replaced with new identifiers
- 5185     • urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address
- 5186         • Replaced with urn:oasis:names:tc:xacml:3.0:subject:authn-locality:ip-  
5187 address
- 5188     • urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name
- 5189         • Replaced with urn:oasis:names:tc:xacml:3.0:subject:authn-  
5190 locality:dns-name
- 5191
- 5192 The following combining algorithms have been replaced with new variants which allow for better handling  
5193 of "Indeterminate" results.
- 5194     • urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides
- 5195         • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides
- 5196     • urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides
- 5197         • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides
- 5198     • urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides
- 5199         • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides
- 5200     • urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides
- 5201         • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides
- 5202     • urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides
- 5203         • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides
- 5204     • urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides
- 5205         • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-overrides
- 5206     • urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-overrides
- 5207         • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-overrides
- 5208     • urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-overrides
- 5209         • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-overrides

---

## 5210 Appendix B. XACML identifiers (normative)

5211 This section defines standard identifiers for commonly used entities.

### 5212 B.1 XACML namespaces

5213 XACML is defined using this identifier.

5214 `urn:oasis:names:tc:xacml:3.0:core:schema`

### 5215 B.2 Attribute categories

5216 The following **attribute** category identifiers MUST be used when an XACML 2.0 or earlier **policy** or  
5217 request is translated into XACML 3.0.

5218 **Attributes** previously placed in the **Resource**, **Action**, and **Environment** sections of a request are  
5219 placed in an **attribute** category with the following identifiers respectively. It is RECOMMENDED that they  
5220 are used to list **attributes** of **resources**, **actions**, and the **environment** respectively when authoring  
5221 XACML 3.0 **policies** or requests.

5222 `urn:oasis:names:tc:xacml:3.0:attribute-category:resource`

5223 `urn:oasis:names:tc:xacml:3.0:attribute-category:action`

5224 `urn:oasis:names:tc:xacml:3.0:attribute-category:environment`

5225 **Attributes** previously placed in the **Subject** section of a request are placed in an **attribute** category  
5226 which is identical of the **subject** category in XACML 2.0, as defined below. It is RECOMMENDED that  
5227 they are used to list **attributes** of **subjects** when authoring XACML 3.0 **policies** or requests.

5228 This identifier indicates the system entity that initiated the **access** request. That is, the initial entity in a  
5229 request chain. If **subject** category is not specified in XACML 2.0, this is the default translation value.

5230 `urn:oasis:names:tc:xacml:1.0:subject-category:access-subject`

5231 This identifier indicates the system entity that will receive the results of the request (used when it is  
5232 distinct from the **access-subject**).

5233 `urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject`

5234 This identifier indicates a system entity through which the **access** request was passed.

5235 `urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject`

5236 This identifier indicates a system entity associated with a local or remote codebase that generated the  
5237 request. Corresponding **subject attributes** might include the URL from which it was loaded and/or the  
5238 identity of the code-signer.

5239 `urn:oasis:names:tc:xacml:1.0:subject-category:codebase`

5240 This identifier indicates a system entity associated with the computer that initiated the **access** request.  
5241 An example would be an IPsec identity.

5242 `urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine`

### 5243 B.3 Data-types

5244 The following identifiers indicate data-types that are defined in Section A.2.

5245 `urn:oasis:names:tc:xacml:1.0:data-type:x500Name`

5246 `urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name`

5247 `urn:oasis:names:tc:xacml:2.0:data-type:ipAddress`

5248 `urn:oasis:names:tc:xacml:2.0:data-type:dnsName`

5249 `urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression`

5250 The following data-type identifiers are defined by XML Schema [XS].  
5251 <http://www.w3.org/2001/XMLSchema#string>  
5252 <http://www.w3.org/2001/XMLSchema#boolean>  
5253 <http://www.w3.org/2001/XMLSchema#integer>  
5254 <http://www.w3.org/2001/XMLSchema#double>  
5255 <http://www.w3.org/2001/XMLSchema#time>  
5256 <http://www.w3.org/2001/XMLSchema#date>  
5257 <http://www.w3.org/2001/XMLSchema#dateTime>  
5258 <http://www.w3.org/2001/XMLSchema#anyURI>  
5259 <http://www.w3.org/2001/XMLSchema#hexBinary>  
5260 <http://www.w3.org/2001/XMLSchema#base64Binary>  
5261 The following data-type identifiers correspond to the `dayTimeDuration` and `yearMonthDuration` data-types  
5262 defined in [XF] Sections 10.3.2 and 10.3.1, respectively.  
5263 <http://www.w3.org/2001/XMLSchema#dayTimeDuration>  
5264 <http://www.w3.org/2001/XMLSchema#yearMonthDuration>

## 5265 B.4 Subject attributes

5266 These identifiers indicate **attributes** of a **subject**. When used, it is RECOMMENDED that they appear  
5267 within an `<Attributes>` element of the request **context** with a **subject** category (see section B.2).  
5268 At most one of each of these **attributes** is associated with each **subject**. Each **attribute** associated with  
5269 authentication included within a single `<Attributes>` element relates to the same authentication event.  
5270 This identifier indicates the name of the **subject**.  
5271 `urn:oasis:names:tc:xacml:1.0:subject:subject-id`  
5272 This identifier indicates the security domain of the subject. It identifies the administrator and **policy** that  
5273 manages the name-space in which the **subject** id is administered.  
5274 `urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier`  
5275 This identifier indicates a public key used to confirm the **subject's** identity.  
5276 `urn:oasis:names:tc:xacml:1.0:subject:key-info`  
5277 This identifier indicates the time at which the **subject** was authenticated.  
5278 `urn:oasis:names:tc:xacml:1.0:subject:authentication-time`  
5279 This identifier indicates the method used to authenticate the **subject**.  
5280 `urn:oasis:names:tc:xacml:1.0:subject:authentication-method`  
5281 This identifier indicates the time at which the **subject** initiated the **access** request, according to the **PEP**.  
5282 `urn:oasis:names:tc:xacml:1.0:subject:request-time`  
5283 This identifier indicates the time at which the **subject's** current session began, according to the **PEP**.  
5284 `urn:oasis:names:tc:xacml:1.0:subject:session-start-time`  
5285 The following identifiers indicate the location where authentication credentials were activated.  
5286 This identifier indicates that the location is expressed as an IP address.  
5287 `urn:oasis:names:tc:xacml:3.0:subject:authn-locality:ip-address`  
5288 The corresponding **attribute** SHALL be of data-type "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress".  
5289 This identifier indicates that the location is expressed as a DNS name.  
5290 `urn:oasis:names:tc:xacml:3.0:subject:authn-locality:dns-name`  
5291 The corresponding **attribute** SHALL be of data-type "urn:oasis:names:tc:xacml:2.0:data-type:dnsName".

5292 Where a suitable **attribute** is already defined in LDAP [LDAP-1], [LDAP-2], the XACML identifier SHALL  
5293 be formed by adding the **attribute** name to the URI of the LDAP specification. For example, the **attribute**  
5294 name for the userPassword defined in the RFC 2256 SHALL be:  
5295 `http://www.ietf.org/rfc/rfc2256.txt#userPassword`

## 5296 B.5 Resource attributes

5297 These identifiers indicate **attributes** of the **resource**. When used, it is RECOMMENDED they appear  
5298 within the <Attributes> element of the request **context** with *Category*  
5299 `urn:oasis:names:tc:xacml:3.0:attribute-category:resource`.

5300 This **attribute** identifies the **resource** to which **access** is requested.

5301 `urn:oasis:names:tc:xacml:1.0:resource:resource-id`

5302 This **attribute** identifies the namespace of the top element(s) of the contents of the <Content> element.  
5303 In the case where the **resource** content is supplied in the request **context** and the **resource**  
5304 namespaces are defined in the **resource**, the **PEP** MAY provide this **attribute** in the request to indicate  
5305 the namespaces of the **resource** content. In this case there SHALL be one value of this **attribute** for  
5306 each unique namespace of the top level elements in the <Content> element. The type of the  
5307 corresponding **attribute** SHALL be "http://www.w3.org/2001/XMLSchema#anyURI".

5308 `urn:oasis:names:tc:xacml:2.0:resource:target-namespace`

## 5309 B.6 Action attributes

5310 These identifiers indicate **attributes** of the **action** being requested. When used, it is RECOMMENDED  
5311 they appear within the <Attributes> element of the request **context** with *Category*  
5312 `urn:oasis:names:tc:xacml:3.0:attribute-category:action`.

5313 This **attribute** identifies the **action** for which **access** is requested.

5314 `urn:oasis:names:tc:xacml:1.0:action:action-id`

5315 Where the **action** is implicit, the value of the action-id **attribute** SHALL be

5316 `urn:oasis:names:tc:xacml:1.0:action:implied-action`

5317 This **attribute** identifies the namespace in which the action-id **attribute** is defined.

5318 `urn:oasis:names:tc:xacml:1.0:action:action-namespace`

## 5319 B.7 Environment attributes

5320 These identifiers indicate **attributes** of the **environment** within which the **decision request** is to be  
5321 evaluated. When used in the **decision request**, it is RECOMMENDED they appear in the  
5322 <Attributes> element of the request **context** with *Category* `urn:oasis:names:tc:xacml:3.0:attribute-`  
5323 `category:environment`.

5324 This identifier indicates the current time at the **context handler**. In practice it is the time at which the  
5325 request **context** was created. For this reason, if these identifiers appear in multiple places within a  
5326 <Policy> or <PolicySet>, then the same value SHALL be assigned to each occurrence in the  
5327 evaluation procedure, regardless of how much time elapses between the processing of the occurrences.

5328 `urn:oasis:names:tc:xacml:1.0:environment:current-time`

5329 The corresponding **attribute** SHALL be of data-type "http://www.w3.org/2001/XMLSchema#time".

5330 `urn:oasis:names:tc:xacml:1.0:environment:current-date`

5331 The corresponding **attribute** SHALL be of data-type "http://www.w3.org/2001/XMLSchema#date".

5332 `urn:oasis:names:tc:xacml:1.0:environment:current-dateTime`

5333 The corresponding **attribute** SHALL be of data-type "http://www.w3.org/2001/XMLSchema#dateTime".

## 5334 B.8 Status codes

5335 The following status code values are defined.

5336 This identifier indicates success.

5337 urn:oasis:names:tc:xacml:1.0:status:ok

5338 This identifier indicates that all the **attributes** necessary to make a **policy decision** were not available (see Section 5.58).

5340 urn:oasis:names:tc:xacml:1.0:status:missing-attribute

5341 This identifier indicates that some **attribute** value contained a syntax error, such as a letter in a numeric field.

5343 urn:oasis:names:tc:xacml:1.0:status:syntax-error

5344 This identifier indicates that an error occurred during **policy** evaluation. An example would be division by zero.

5346 urn:oasis:names:tc:xacml:1.0:status:processing-error

## 5347 B.9 Combining algorithms

5348 The deny-overrides **rule-combining algorithm** has the following value for the `ruleCombiningAlgId` attribute:

5350 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides

5351 The deny-overrides **policy-combining algorithm** has the following value for the `policyCombiningAlgId` attribute:

5353 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides

5354 The permit-overrides **rule-combining algorithm** has the following value for the `ruleCombiningAlgId` attribute:

5356 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides

5357 The permit-overrides **policy-combining algorithm** has the following value for the `policyCombiningAlgId` attribute:

5359 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides

5360 The first-applicable **rule-combining algorithm** has the following value for the `ruleCombiningAlgId` attribute:

5362 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable

5363 The first-applicable **policy-combining algorithm** has the following value for the `policyCombiningAlgId` attribute:

5365 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable

5366 The only-one-applicable-policy **policy-combining algorithm** has the following value for the `policyCombiningAlgId` attribute:

5368 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable

5369 The ordered-deny-overrides **rule-combining algorithm** has the following value for the `ruleCombiningAlgId` attribute:

5371 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides

5372 The ordered-deny-overrides **policy-combining algorithm** has the following value for the `policyCombiningAlgId` attribute:

5374 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-overrides

5376 The ordered-permit-overrides **rule-combining algorithm** has the following value for the `ruleCombiningAlgId` attribute:

5378 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-  
5379 overrides  
5380 The ordered-permit-overrides **policy-combining algorithm** has the following value for the  
5381 policyCombiningAlgId attribute:  
5382 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-  
5383 overrides  
5384 The deny-unless-permit **rule-combining algorithm** has the following value for the  
5385 policyCombiningAlgId attribute:  
5386 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit  
5387 The permit-unless-deny **rule-combining algorithm** has the following value for the  
5388 policyCombiningAlgId attribute:  
5389 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-unless-deny  
5390 The deny-unless-permit **policy-combining algorithm** has the following value for the  
5391 policyCombiningAlgId attribute:  
5392 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit  
5393 The permit-unless-deny **policy-combining algorithm** has the following value for the  
5394 policyCombiningAlgId attribute:  
5395 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-unless-deny  
5396 The legacy deny-overrides **rule-combining algorithm** has the following value for the  
5397 ruleCombiningAlgId attribute:  
5398 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides  
5399 The legacy deny-overrides **policy-combining algorithm** has the following value for the  
5400 policyCombiningAlgId attribute:  
5401 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides  
5402 The legacy permit-overrides **rule-combining algorithm** has the following value for the  
5403 ruleCombiningAlgId attribute:  
5404 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides  
5405 The legacy permit-overrides **policy-combining algorithm** has the following value for the  
5406 policyCombiningAlgId attribute:  
5407 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides  
5408 The legacy ordered-deny-overrides **rule-combining algorithm** has the following value for the  
5409 ruleCombiningAlgId attribute:  
5410 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides  
5411 The legacy ordered-deny-overrides **policy-combining algorithm** has the following value for the  
5412 policyCombiningAlgId attribute:  
5413 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-  
5414 overrides  
5415 The legacy ordered-permit-overrides **rule-combining algorithm** has the following value for the  
5416 ruleCombiningAlgId attribute:  
5417 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-  
5418 overrides  
5419 The legacy ordered-permit-overrides **policy-combining algorithm** has the following value for the  
5420 policyCombiningAlgId attribute:  
5421 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-  
5422 overrides  
5423

## 5424 Appendix C. Combining algorithms (normative)

5425 This section contains a description of the *rule-* and *policy-combining algorithms* specified by XACML.  
5426 Pseudo code is normative, descriptions in English are non-normative.

5427 The legacy *combining algorithms* are defined in previous versions of XACML, and are retained for  
5428 compatibility reasons. It is RECOMMENDED that the new *combining algorithms* are used instead of the  
5429 legacy *combining algorithms* for new use.

5430 Note that in each case an implementation is conformant as long as it produces the same result as is  
5431 specified here, regardless of how and in what order the implementation behaves internally.

### 5432 C.1 Extended Indeterminate values

5433 Some combining algorithms are defined in terms of an extended set of "Indeterminate" values. See  
5434 section 7.10 for the definition of the Extended Indeterminate values. For these algorithms, the *PDP* MUST  
5435 keep track of the extended set of "Indeterminate" values during *rule* and *policy* combining.

5436 The output of a combining algorithm which does not track the extended set of "Indeterminate" values  
5437 MUST be treated as "Indeterminate{DP}" for the value "Indeterminate" by a combining algorithm which  
5438 tracks the extended set of "Indeterminate" values.

5439 A combining algorithm which does not track the extended set of "Indeterminate" values MUST treat the  
5440 output of a combining algorithm which tracks the extended set of "Indeterminate" values as an  
5441 "Indeterminate" for any of the possible values of the extended set of "Indeterminate".

### 5442 C.2 Deny-overrides

5443 This section defines the "Deny-overrides" *rule-combining algorithm* of a *policy* and *policy-combining*  
5444 *algorithm* of a *policy set*.

5445 This *combining algorithm* makes use of the extended "Indeterminate".

5446 The *rule combining algorithm* defined here has the following identifier:

5447 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides

5448 The *policy combining algorithm* defined here has the following identifier:

5449 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides

5450 The following is a non-normative informative description of this *combining algorithm*.

5451 The deny overrides *combining algorithm* is intended for those cases where a deny  
5452 decision should have priority over a permit decision. This algorithm has the following  
5453 behavior.

- 5454 1. If any decision is "Deny", the result is "Deny".
- 5455 2. Otherwise, if any decision is "Indeterminate{DP}", the result is "Indeterminate{DP}".
- 5456 3. Otherwise, if any decision is "Indeterminate{D}" and another decision is "Indeterminate{P} or  
5457 Permit, the result is "Indeterminate{DP}".
- 5458 4. Otherwise, if any decision is "Indeterminate{D}", the result is "Indeterminate{D}".
- 5459 5. Otherwise, if any decision is "Permit", the result is "Permit".
- 5460 6. Otherwise, if any decision is "Indeterminate{P}", the result is "Indeterminate{P}".
- 5461 7. Otherwise, the result is "NotApplicable".

5462 The following pseudo-code represents the normative specification of this *combining algorithm*. The  
5463 algorithm is presented here in a form where the input to it is an array with children (the *policies*, *policy*  
5464 *sets* or *rules*) of the *policy* or *policy set*. The children may be processed in any order, so the set of  
5465 obligations or advice provided by this algorithm is not deterministic.

```

5466 Decision denyOverridesCombiningAlgorithm(Node[] children)
5467 {
5468     Boolean atLeastOneErrorD = false;
5469     Boolean atLeastOneErrorP = false;
5470     Boolean atLeastOneErrorDP = false;
5471     Boolean atLeastOnePermit = false;
5472     for( i=0 ; i < lengthOf(children) ; i++ )
5473     {
5474         Decision decision = children[i].evaluate();
5475         if (decision == Deny)
5476         {
5477             return Deny;
5478         }
5479         if (decision == Permit)
5480         {
5481             atLeastOnePermit = true;
5482             continue;
5483         }
5484         if (decision == NotApplicable)
5485         {
5486             continue;
5487         }
5488         if (decision == Indeterminate{D})
5489         {
5490             atLeastOneErrorD = true;
5491             continue;
5492         }
5493         if (decision == Indeterminate{P})
5494         {
5495             atLeastOneErrorP = true;
5496             continue;
5497         }
5498         if (decision == Indeterminate{DP})
5499         {
5500             atLeastOneErrorDP = true;
5501             continue;
5502         }
5503     }
5504     if (atLeastOneErrorDP)
5505     {
5506         return Indeterminate{DP};
5507     }
5508     if (atLeastOneErrorD && (atLeastOneErrorP || atLeastOnePermit))
5509     {
5510         return Indeterminate{DP};
5511     }
5512     if (atLeastOneErrorD)
5513     {
5514         return Indeterminate{D};
5515     }
5516     if (atLeastOnePermit)
5517     {
5518         return Permit;
5519     }
5520     if (atLeastOneErrorP)
5521     {
5522         return Indeterminate{P};
5523     }
5524     return NotApplicable;
5525 }

```

5526 **Obligations** and **advice** shall be combined as described in Section 7.18.



### 5527 C.3 Ordered-deny-overrides

5528 The following specification defines the "Ordered-deny-overrides" **rule-combining algorithm** of a **policy**.

5529 The behavior of this algorithm is identical to that of the "Deny-overrides" **rule-combining**  
5530 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL  
5531 match the order as listed in the **policy**.

5532 The **rule combining algorithm** defined here has the following identifier:

5533 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides

5534 The following specification defines the "Ordered-deny-overrides" **policy-combining algorithm** of a  
5535 **policy set**.

5536 The behavior of this algorithm is identical to that of the "Deny-overrides" **policy-combining**  
5537 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL  
5538 match the order as listed in the **policy set**.

5539 The **policy combining algorithm** defined here has the following identifier:

5540 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-  
5541 overrides

### 5542 C.4 Permit-overrides

5543 This section defines the "Permit-overrides" **rule-combining algorithm** of a **policy** and **policy-combining**  
5544 **algorithm** of a **policy set**.

5545 This **combining algorithm** makes use of the extended "Indeterminate".

5546 The **rule combining algorithm** defined here has the following identifier:

5547 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides

5548 The **policy combining algorithm** defined here has the following identifier:

5549 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides

5550 The following is a non-normative informative description of this combining algorithm.

5551 The permit overrides **combining algorithm** is intended for those cases where a permit  
5552 decision should have priority over a deny decision. This algorithm has the following  
5553 behavior.

- 5554 1. If any decision is "Permit", the result is "Permit".
- 5555 2. Otherwise, if any decision is "Indeterminate{DP}", the result is "Indeterminate{DP}".
- 5556 3. Otherwise, if any decision is "Indeterminate{P}" and another decision is  
5557 "Indeterminate{D} or Deny, the result is "Indeterminate{DP}".
- 5558 4. Otherwise, if any decision is "Indeterminate{P}", the result is "Indeterminate{P}".
- 5559 5. Otherwise, if any decision is "Deny", the result is "Deny".
- 5560 6. Otherwise, if any decision is "Indeterminate{D}", the result is "Indeterminate{D}".
- 5561 7. Otherwise, the result is "NotApplicable".

5562 The following pseudo-code represents the normative specification of this **combining algorithm**. The  
5563 algorithm is presented here in a form where the input to it is an array with all children (the **policies**, **policy**  
5564 **sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set of  
5565 obligations or advice provided by this algorithm is not deterministic.

```
5566 Decision permitOverridesCombiningAlgorithm(Node[] children)  
5567 {  
5568     Boolean atLeastOneErrorD = false;  
5569     Boolean atLeastOneErrorP = false;  
5570     Boolean atLeastOneErrorDP = false;  
5571     Boolean atLeastOneDeny = false;
```

```

5572 for( i=0 ; i < lengthOf(children) ; i++ )
5573 {
5574     Decision decision = children[i].evaluate();
5575     if (decision == Deny)
5576     {
5577         atLeastOneDeny = true;
5578         continue;
5579     }
5580     if (decision == Permit)
5581     {
5582         return Permit;
5583     }
5584     if (decision == NotApplicable)
5585     {
5586         continue;
5587     }
5588     if (decision == Indeterminate{D})
5589     {
5590         atLeastOneErrorD = true;
5591         continue;
5592     }
5593     if (decision == Indeterminate{P})
5594     {
5595         atLeastOneErrorP = true;
5596         continue;
5597     }
5598     if (decision == Indeterminate{DP})
5599     {
5600         atLeastOneErrorDP = true;
5601         continue;
5602     }
5603 }
5604 if (atLeastOneErrorDP)
5605 {
5606     return Indeterminate{DP};
5607 }
5608 if (atLeastOneErrorP && (atLeastOneErrorD || atLeastOneDeny))
5609 {
5610     return Indeterminate{DP};
5611 }
5612 if (atLeastOneErrorP)
5613 {
5614     return Indeterminate{P};
5615 }
5616 if (atLeastOneDeny)
5617 {
5618     return Deny;
5619 }
5620 if (atLeastOneErrorD)
5621 {
5622     return Indeterminate{D};
5623 }
5624 return NotApplicable;
5625 }

```

5626 **Obligations** and **advice** shall be combined as described in Section 7.18.

## 5627 C.5 Ordered-permit-overrides

5628 The following specification defines the "Ordered-permit-overrides" **rule-combining algorithm** of a **policy**.

5629 The behavior of this algorithm is identical to that of the "Permit-overrides" **rule-combining**  
5630 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL  
5631 match the order as listed in the **policy**.

5632 The **rule combining algorithm** defined here has the following identifier:  
5633 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-  
5634 overrides

5635 The following specification defines the "Ordered-permit-overrides" **policy-combining algorithm** of a  
5636 **policy set**.

5637 The behavior of this algorithm is identical to that of the "Permit-overrides" **policy-combining**  
5638 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL  
5639 match the order as listed in the **policy set**.

5640 The **policy combining algorithm** defined here has the following identifier:  
5641 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-  
5642 overrides

## 5643 C.6 Deny-unless-permit

5644 This section defines the "Deny-unless-permit" **rule-combining algorithm** of a **policy** or **policy-**  
5645 **combining algorithm** of a **policy set**.

5646 The **rule combining algorithm** defined here has the following identifier:  
5647 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit

5648 The **policy combining algorithm** defined here has the following identifier:  
5649 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit

5650 The following is a non-normative informative description of this **combining algorithm**.

5651 The "Deny-unless-permit" **combining algorithm** is intended for those cases where a  
5652 permit decision should have priority over a deny decision, and an "Indeterminate" or  
5653 "NotApplicable" must never be the result. It is particularly useful at the top level in a  
5654 **policy** structure to ensure that a **PDP** will always return a definite "Permit" or "Deny"  
5655 result. This algorithm has the following behavior.

- 5656 1. If any decision is "Permit", the result is "Permit".
- 5657 2. Otherwise, the result is "Deny".

5658 The following pseudo-code represents the normative specification of this **combining algorithm**. The  
5659 algorithm is presented here in a form where the input to it is an array with all the children (the **policies**,  
5660 **policy sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set  
5661 of obligations or advice provided by this algorithm is not deterministic.

```
5662 Decision denyUnlessPermitCombiningAlgorithm(Node[] children)  
5663 {  
5664     for( i=0 ; i < lengthOf(children) ; i++ )  
5665     {  
5666         if (children[i].evaluate() == Permit)  
5667         {  
5668             return Permit;  
5669         }  
5670     }  
5671     return Deny;  
5672 }
```

5673 **Obligations** and **advice** shall be combined as described in Section 7.18.

## 5674 C.7 Permit-unless-deny

5675 This section defines the "Permit-unless-deny" **rule-combining algorithm** of a **policy** or **policy-**  
5676 **combining algorithm** of a **policy set**.

5677 The **rule combining algorithm** defined here has the following identifier:  
5678 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-unless-deny

5679 The **policy combining algorithm** defined here has the following identifier:  
5680 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-unless-deny  
5681 The following is a non-normative informative description of this **combining algorithm**.

5682 The "Permit-unless-deny" **combining algorithm** is intended for those cases where a  
5683 deny decision should have priority over a permit decision, and an "Indeterminate" or  
5684 "NotApplicable" must never be the result. It is particularly useful at the top level in a  
5685 **policy** structure to ensure that a **PDP** will always return a definite "Permit" or "Deny"  
5686 result. This algorithm has the following behavior.

- 5687 1. If any decision is "Deny", the result is "Deny".
- 5688 2. Otherwise, the result is "Permit".

5689 The following pseudo-code represents the normative specification of this **combining algorithm**. The  
5690 algorithm is presented here in a form where the input to it is an array with all the children (the **policies**,  
5691 **policy sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set  
5692 of obligations or advice provided by this algorithm is not deterministic.

```
5693 Decision permitUnlessDenyCombiningAlgorithm(Node[] children)
5694 {
5695   for( i=0 ; i < lengthOf(children) ; i++ )
5696   {
5697     if (children[i].evaluate() == Deny)
5698     {
5699       return Deny;
5700     }
5701   }
5702   return Permit;
5703 }
```

5704 **Obligations** and **advice** shall be combined as described in Section 7.18.

## 5705 C.8 First-applicable

5706 This section defines the "First-applicable" **rule-combining algorithm** of a **policy** and **policy-combining**  
5707 **algorithm** of a **policy set**.

5708 The **rule combining algorithm** defined here has the following identifier:

5709 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable

5710 The following is a non-normative informative description of the "First-Applicable" **rule-combining**  
5711 **algorithm** of a **policy**.

5712 Each **rule** SHALL be evaluated in the order in which it is listed in the **policy**. For a particular  
5713 **rule**, if the **target** matches and the **condition** evaluates to "True", then the evaluation of the  
5714 **policy** SHALL halt and the corresponding **effect** of the **rule** SHALL be the result of the evaluation  
5715 of the **policy** (i.e. "Permit" or "Deny"). For a particular **rule** selected in the evaluation, if the  
5716 **target** evaluates to "False" or the **condition** evaluates to "False", then the next **rule** in the order  
5717 SHALL be evaluated. If no further **rule** in the order exists, then the **policy** SHALL evaluate to  
5718 "NotApplicable".

5719 If an error occurs while evaluating the **target** or **condition** of a **rule**, then the evaluation SHALL  
5720 halt, and the **policy** shall evaluate to "Indeterminate", with the appropriate error status.

5721 The following pseudo-code represents the normative specification of this **rule-combining algorithm**.

```
5722 Decision firstApplicableEffectRuleCombiningAlgorithm(Rule[] rules)
5723 {
5724   for( i = 0 ; i < lengthOf(rules) ; i++ )
5725   {
5726     Decision decision = evaluate(rules[i]);
5727     if (decision == Deny)
5728     {
```

```

5729         return Deny;
5730     }
5731     if (decision == Permit)
5732     {
5733         return Permit;
5734     }
5735     if (decision == NotApplicable)
5736     {
5737         continue;
5738     }
5739     if (decision == Indeterminate)
5740     {
5741         return Indeterminate;
5742     }
5743 }
5744 return NotApplicable;
5745 }

```

5746 The **policy combining algorithm** defined here has the following identifier:

5747 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable

5748 The following is a non-normative informative description of the "First-applicable" **policy-combining algorithm** of a **policy set**.

5750 Each **policy** is evaluated in the order that it appears in the **policy set**. For a particular **policy**, if  
5751 the **target** evaluates to "True" and the **policy** evaluates to a determinate value of "Permit" or  
5752 "Deny", then the evaluation SHALL halt and the **policy set** SHALL evaluate to the **effect** value of  
5753 that **policy**. For a particular **policy**, if the **target** evaluate to "False", or the **policy** evaluates to  
5754 "NotApplicable", then the next **policy** in the order SHALL be evaluated. If no further **policy** exists  
5755 in the order, then the **policy set** SHALL evaluate to "NotApplicable".

5756 If an error were to occur when evaluating the **target**, or when evaluating a specific **policy**, the  
5757 reference to the **policy** is considered invalid, or the **policy** itself evaluates to "Indeterminate",  
5758 then the evaluation of the **policy-combining algorithm** shall halt, and the **policy set** shall  
5759 evaluate to "Indeterminate" with an appropriate error status.

5760 The following pseudo-code represents the normative specification of this policy-combination algorithm.

```

5761 Decision firstApplicableEffectPolicyCombiningAlgorithm(Policy[] policies)
5762 {
5763     for( i = 0 ; i < lengthOf(policies) ; i++ )
5764     {
5765         Decision decision = evaluate(policies[i]);
5766         if(decision == Deny)
5767         {
5768             return Deny;
5769         }
5770         if(decision == Permit)
5771         {
5772             return Permit;
5773         }
5774         if (decision == NotApplicable)
5775         {
5776             continue;
5777         }
5778         if (decision == Indeterminate)
5779         {
5780             return Indeterminate;
5781         }
5782     }
5783     return NotApplicable;
5784 }

```

5785 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

## 5786 C.9 Only-one-applicable

5787 This section defines the "Only-one-applicable" **policy-combining algorithm** of a **policy set**.

5788 The **policy combining algorithm** defined here has the following identifier:

5789 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable

5790 The following is a non-normative informative description of the "Only-one-applicable" **policy-combining algorithm** of a **policy set**.

5792 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".

5796 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**.

5798 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.

5801 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```
5802 Decision onlyOneApplicablePolicyPolicyCombiningAlgorithm(Policy[] policies)
5803 {
5804     Boolean          atLeastOne      = false;
5805     Policy           selectedPolicy = null;
5806     ApplicableResult appResult;
5807
5808     for ( i = 0; i < lengthOf(policies) ; i++ )
5809     {
5810         appResult = isApplicable(policies[i]);
5811
5812         if ( appResult == Indeterminate )
5813         {
5814             return Indeterminate;
5815         }
5816         if( appResult == Applicable )
5817         {
5818             if ( atLeastOne )
5819             {
5820                 return Indeterminate;
5821             }
5822             else
5823             {
5824                 atLeastOne      = true;
5825                 selectedPolicy = policies[i];
5826             }
5827         }
5828         if ( appResult == NotApplicable )
5829         {
5830             continue;
5831         }
5832     }
5833     if ( atLeastOne )
5834     {
5835         return evaluate(selectedPolicy);
5836     }
5837     else
5838     {
5839         return NotApplicable;
5840     }
5841 }
```

5842 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

## 5843 C.10 Legacy Deny-overrides

5844 This section defines the legacy "Deny-overrides" *rule-combining algorithm* of a *policy* and *policy-*  
5845 *combining algorithm* of a *policy set*.

5846

5847 The *rule combining algorithm* defined here has the following identifier:

5848 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides

5849 The following is a non-normative informative description of this combining algorithm.

5850 The "Deny-overrides" rule combining algorithm is intended for those cases where a deny  
5851 decision should have priority over a permit decision. This algorithm has the following  
5852 behavior.

- 5853 1. If any rule evaluates to "Deny", the result is "Deny".
- 5854 2. Otherwise, if any rule having Effect="Deny" evaluates to "Indeterminate", the result is  
5855 "Indeterminate".
- 5856 3. Otherwise, if any rule evaluates to "Permit", the result is "Permit".
- 5857 4. Otherwise, if any rule having Effect="Permit" evaluates to "Indeterminate", the result is  
5858 "Indeterminate".
- 5859 5. Otherwise, the result is "NotApplicable".

5860 The following pseudo-code represents the normative specification of this *rule-combining algorithm*.

```
5861 Decision denyOverridesRuleCombiningAlgorithm(Rule[] rules)
5862 {
5863     Boolean atLeastOneError = false;
5864     Boolean potentialDeny = false;
5865     Boolean atLeastOnePermit = false;
5866     for( i=0 ; i < lengthOf(rules) ; i++ )
5867     {
5868         Decision decision = evaluate(rules[i]);
5869         if (decision == Deny)
5870         {
5871             return Deny;
5872         }
5873         if (decision == Permit)
5874         {
5875             atLeastOnePermit = true;
5876             continue;
5877         }
5878         if (decision == NotApplicable)
5879         {
5880             continue;
5881         }
5882         if (decision == Indeterminate)
5883         {
5884             atLeastOneError = true;
5885
5886             if (effect(rules[i]) == Deny)
5887             {
5888                 potentialDeny = true;
5889             }
5890             continue;
5891         }
5892     }
5893     if (potentialDeny)
5894     {
5895         return Indeterminate;
5896     }
5897     if (atLeastOnePermit)
5898     {
```

```
5899     return Permit;
5900   }
5901   if (atLeastOneError)
5902   {
5903     return Indeterminate;
5904   }
5905   return NotApplicable;
5906 }
```

5907 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

5908 The **policy combining algorithm** defined here has the following identifier:

5909 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides

5910 The following is a non-normative informative description of this combining algorithm.

5911 The "Deny-overrides" policy combining algorithm is intended for those cases where a  
5912 deny decision should have priority over a permit decision. This algorithm has the  
5913 following behavior.

- 5914 1. If any policy evaluates to "Deny", the result is "Deny".
- 5915 2. Otherwise, if any policy evaluates to "Indeterminate", the result is "Deny".
- 5916 3. Otherwise, if any policy evaluates to "Permit", the result is "Permit".
- 5917 4. Otherwise, the result is "NotApplicable".

5918 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```
5919 Decision denyOverridesPolicyCombiningAlgorithm(Policy[] policies)
5920 {
5921   Boolean atLeastOnePermit = false;
5922   for( i=0 ; i < lengthOf(policies) ; i++ )
5923   {
5924     Decision decision = evaluate(policies[i]);
5925     if (decision == Deny)
5926     {
5927       return Deny;
5928     }
5929     if (decision == Permit)
5930     {
5931       atLeastOnePermit = true;
5932       continue;
5933     }
5934     if (decision == NotApplicable)
5935     {
5936       continue;
5937     }
5938     if (decision == Indeterminate)
5939     {
5940       return Deny;
5941     }
5942   }
5943   if (atLeastOnePermit)
5944   {
5945     return Permit;
5946   }
5947   return NotApplicable;
5948 }
```

5949 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

## 5950 C.11 Legacy Ordered-deny-overrides

5951 The following specification defines the legacy "Ordered-deny-overrides" **rule-combining algorithm** of a  
5952 **policy**.



5953 The behavior of this algorithm is identical to that of the "Deny-overrides" **rule-combining**  
5954 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL  
5955 match the order as listed in the **policy**.

5956 The **rule combining algorithm** defined here has the following identifier:

5957 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides

5958 The following specification defines the legacy "Ordered-deny-overrides" **policy-combining algorithm** of  
5959 a **policy set**.

5960 The behavior of this algorithm is identical to that of the "Deny-overrides" **policy-combining**  
5961 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL  
5962 match the order as listed in the **policy set**.

5963 The **rule combining algorithm** defined here has the following identifier:

5964 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-  
5965 overrides

## 5966 C.12 Legacy Permit-overrides

5967 This section defines the legacy "Permit-overrides" **rule-combining algorithm** of a **policy** and **policy-**  
5968 **combining algorithm** of a **policy set**.

5969 The **rule combining algorithm** defined here has the following identifier:

5970 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides

5971 The following is a non-normative informative description of this combining algorithm.

5972 The "Permit-overrides" rule combining algorithm is intended for those cases where a  
5973 permit decision should have priority over a deny decision. This algorithm has the  
5974 following behavior.

- 5975 1. If any rule evaluates to "Permit", the result is "Permit".
- 5976 2. Otherwise, if any rule having Effect="Permit" evaluates to "Indeterminate" the result is  
5977 "Indeterminate".
- 5978 3. Otherwise, if any rule evaluates to "Deny", the result is "Deny".
- 5979 4. Otherwise, if any rule having Effect="Deny" evaluates to "Indeterminate", the result is  
5980 "Indeterminate".
- 5981 5. Otherwise, the result is "NotApplicable".

5982 The following pseudo-code represents the normative specification of this **rule-combining algorithm**.

```
5983 Decision permitOverridesRuleCombiningAlgorithm(Rule[] rules)
5984 {
5985     Boolean atLeastOneError = false;
5986     Boolean potentialPermit = false;
5987     Boolean atLeastOneDeny = false;
5988     for( i=0 ; i < lengthOf(rules) ; i++ )
5989     {
5990         Decision decision = evaluate(rules[i]);
5991         if (decision == Deny)
5992         {
5993             atLeastOneDeny = true;
5994             continue;
5995         }
5996         if (decision == Permit)
5997         {
5998             return Permit;
5999         }
6000         if (decision == NotApplicable)
6001         {
6002             continue;
6003         }
6004     }
6005 }
```

```

6004     if (decision == Indeterminate)
6005     {
6006         atLeastOneError = true;
6007
6008         if (effect(rules[i]) == Permit)
6009         {
6010             potentialPermit = true;
6011         }
6012         continue;
6013     }
6014 }
6015 if (potentialPermit)
6016 {
6017     return Indeterminate;
6018 }
6019 if (atLeastOneDeny)
6020 {
6021     return Deny;
6022 }
6023 if (atLeastOneError)
6024 {
6025     return Indeterminate;
6026 }
6027 return NotApplicable;
6028 }

```

6029 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

6030 The **policy combining algorithm** defined here has the following identifier:

6031 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides

6032 The following is a non-normative informative description of this combining algorithm.

6033 The "Permit-overrides" policy combining algorithm is intended for those cases where a  
6034 permit decision should have priority over a deny decision. This algorithm has the  
6035 following behavior.

- 6036 1. If any policy evaluates to "Permit", the result is "Permit".
- 6037 2. Otherwise, if any policy evaluates to "Deny", the result is "Deny".
- 6038 3. Otherwise, if any policy evaluates to "Indeterminate", the result is "Indeterminate".
- 6039 4. Otherwise, the result is "NotApplicable".

6040 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```

6041 Decision permitOverridesPolicyCombiningAlgorithm(Policy[] policies)
6042 {
6043     Boolean atLeastOneError = false;
6044     Boolean atLeastOneDeny = false;
6045     for( i=0 ; i < lengthOF(policies) ; i++ )
6046     {
6047         Decision decision = evaluate(policies[i]);
6048         if (decision == Deny)
6049         {
6050             atLeastOneDeny = true;
6051             continue;
6052         }
6053         if (decision == Permit)
6054         {
6055             return Permit;
6056         }
6057         if (decision == NotApplicable)
6058         {
6059             continue;
6060         }
6061         if (decision == Indeterminate)

```

```

6062     {
6063         atLeastOneError = true;
6064         continue;
6065     }
6066 }
6067 if (atLeastOneDeny)
6068 {
6069     return Deny;
6070 }
6071 if (atLeastOneError)
6072 {
6073     return Indeterminate;
6074 }
6075 return NotApplicable;
6076 }

```

6077 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

### 6078 C.13 Legacy Ordered-permit-overrides

6079 The following specification defines the legacy "Ordered-permit-overrides" **rule-combining algorithm** of a  
6080 **policy**.

6081 The behavior of this algorithm is identical to that of the "Permit-overrides" **rule-combining**  
6082 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL  
6083 match the order as listed in the **policy**.

6084 The **rule combining algorithm** defined here has the following identifier:

6085 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-  
6086 overrides

6087 The following specification defines the legacy "Ordered-permit-overrides" **policy-combining algorithm** of  
6088 a **policy set**.

6089 The behavior of this algorithm is identical to that of the "Permit-overrides" **policy-combining**  
6090 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL  
6091 match the order as listed in the **policy set**.

6092 The **policy combining algorithm** defined here has the following identifier:

6093 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-  
6094 overrides

6095

---

6096 **Appendix D. Acknowledgements**

6097 The following individuals have participated in the creation of this specification and are gratefully  
6098 acknowledged:

6099

- 6100 Anil Saldhana
- 6101 Anil Tappetta
- 6102 Anne Anderson
- 6103 Anthony Nadalin
- 6104 Bill Parducci
- 6105 Craig Forster
- 6106 David Chadwick
- 6107 David Staggs
- 6108 Dilli Arumugam
- 6109 Duane DeCouteau
- 6110 Erik Rissanen
- 6111 Gareth Richards
- 6112 Hal Lockhart
- 6113 Jan Herrmann
- 6114 John Tolbert
- 6115 Ludwig Seitz
- 6116 Michiharu Kudo
- 6117 Naomaru Itoi
- 6118 Paul Tyson
- 6119 Prateek Mishra
- 6120 Rich Levinson
- 6121 Ronald Jacobson
- 6122 Seth Proctor
- 6123 Sridhar Muppidi
- 6124 Tim Moses
- 6125 Vernon Murdoch

6126

## Appendix E. Revision History

6127

6128

Revision	Date	Editor	Changes Made
WD 05	10 Oct 2007	Erik Rissanen	Convert to new OASIS template. Fixed typos and errors.
WD 06	18 May 2008	Erik Rissanen	<p>Added missing MaxDelegationDepth in schema fragments.</p> <p>Added missing urn:oasis:names:tc:xacml:1.0:resource:xpath identifier.</p> <p>Corrected typos on xpaths in the example policies.</p> <p>Removed use of xpointer in the examples.</p> <p>Made the &lt;Content&gt; element the context node of all xpath expressions and introduced categorization of XPathS so they point to a specific &lt;Content&gt; element.</p> <p>Added &lt;Content&gt; element to the policy issuer.</p> <p>Added description of the &lt;PolicyIssuer&gt; element.</p> <p>Updated the schema figure in the introduction to reflect the new AllOf/AnyOf schema.</p> <p>Remove duplicate &lt;CombinerParameters&gt; element in the &lt;Policy&gt; element in the schema.</p> <p>Removed default attributes in the schema. (Version in &lt;Policy(Set)&gt; and MustBePresent in &lt;AttributeDesignator&gt; in &lt;AttributeSelector&gt;)</p> <p>Removed references in section 7.3 to the &lt;Condition&gt; element having a FunctionId attribute.</p> <p>Fixed typos in data type URIs in section A.3.7.</p>
WD 07	3 Nov 2008	Erik Rissanen	<p>Fixed "...data-types:..." typo in conformance section.</p> <p>Removed XML default attribute for IncludeInResult for element &lt;Attribute&gt;. Also added this attribute in the associated schema file.</p> <p>Removed description of non-existing XML attribute "ResourceId" from the element &lt;Result&gt;.</p> <p>Moved the urn:oasis:names:tc:xacml:3.0:function:access-permitted function into here from the delegation profile.</p>

		<p>Updated the daytime and yearmonth duration data types to the W3C defined identifiers.</p> <p>Added &lt;Description&gt; to &lt;Apply&gt;.</p> <p>Added XPath versioning to the request.</p> <p>Added security considerations about denial service and the access-permitted function.</p> <p>Changed &lt;Target&gt; matching so NoMatch has priority over Indeterminate.</p> <p>Fixed multiple typos in identifiers.</p> <p>Lower case incorrect use of "MAY".</p> <p>Misc minor typos.</p> <p>Removed whitespace in example attributes.</p> <p>Removed an incorrect sentence about higher order functions in the definition of the &lt;Function&gt; element.</p> <p>Clarified evaluation of empty or missing targets.</p> <p>Use Unicode codepoint collation for string comparisons.</p> <p>Support multiple arguments in multiply functions.</p> <p>Define Indeterminate result for overflow in integer to double conversion.</p> <p>Simplified descriptions of deny/permit overrides algorithms.</p> <p>Add ipAddress and dnsName into conformance section.</p> <p>Don't refer to IEEE 754 for integer arithmetic.</p> <p>Rephrase indeterminate result for arithmetic functions.</p> <p>Fix typos in examples.</p> <p>Clarify Match evaluation and drop list of example functions which can be used in a Match.</p> <p>Added behavior for circular policy/variable references.</p> <p>Fix obligation enforcement so it refers to PEP bias.</p> <p>Added Version xml attribute to the example policies.</p> <p>Remove requirement for PDP to check the target-namespace resource attribute.</p> <p>Added policy identifier list to the response/request.</p> <p>Added statements about Unicode normalization.</p> <p>Clarified definitions of string functions.</p>
--	--	---

			<p>Added new string functions.</p> <p>Added section on Unicode security issues.</p>
WD 08	5 Feb 2009	Erik Rissanen	<p>Updated Unicode normalization section according to suggestion from W3C working group.</p> <p>Set union functions now may take more than two arguments.</p> <p>Made obligation parameters into runtime expressions.</p> <p>Added new combining algorithms</p> <p>Added security consideration about policy id collisions.</p> <p>Added the &lt;Advice&gt; feature</p> <p>Made obligations mandatory (per the 19<sup>th</sup> Dec 2008 decision of the TC)</p> <p>Made obligations/advice available in rules</p> <p>Changed wording about deprecation</p>
WD 09			<p>Clarified wording about normative/informative in the combining algorithms section.</p> <p>Fixed duplicate variable in comb.algs and cleaned up variable names.</p> <p>Updated the schema to support the new multiple request scheme.</p>
WD 10	19 Mar 2009	Erik Rissanen	<p>Fixed schema for &lt;Request&gt;</p> <p>Fixed typos.</p> <p>Added optional Category to AttributeAssignments in obligations/advice.</p>
WD 11		Erik Rissanen	<p>Cleanups courtesy of John Tolbert.</p> <p>Added Issuer XML attribute to &lt;AttributeAssignment&gt;</p> <p>Fix the XPath expressions in the example policies and requests</p> <p>Fix inconsistencies in the conformance tables.</p> <p>Editorial cleanups.</p>
WD 12	16 Nov 2009	Erik Rissanen	<p>(Now working draft after public review of CD 1)</p> <p>Fix typos</p> <p>Allow element selection in attribute selector.</p> <p>Improve consistency in the use of the terms obligation, advice, and advice/obligation expressions and where they can appear.</p> <p>Fixed inconsistency in PEP bias between sections 5.1 and 7.2.</p> <p>Clarified text in overview of combining algorithms.</p> <p>Relaxed restriction on matching in xpath-node-</p>

			<p>match function.</p> <p>Remove note about XPath expert review.</p> <p>Removed obsolete resource:xpath identifier.</p> <p>Updated reference to XML spec.</p> <p>Defined error behavior for string-substring and uri-substring functions.</p> <p>Reversed the order of the arguments for the following functions: string-starts-with, uri-starts-with, string-ends-with, uri-ends-with, string-contains and uri-contains</p> <p>Renamed functions:</p> <ul style="list-style-type: none"> <li>• uri-starts-with to anyURI-starts-with</li> <li>• uri-ends-with to anyURI-ends-with</li> <li>• uri-contains to anyURI-contains</li> <li>• uri-substring to anyURI-substring</li> </ul> <p>Removed redundant occurrence indicators from RequestType.</p> <p>Don't use "...:os" namespace in examples since this is still just "...:wd-12".</p> <p>Added missing MustBePresent and Version XML attributes in example policies.</p> <p>Added missing ReturnPolicyIdList and IncludeInResult XML attributes in example requests.</p> <p>Clarified error behavior in obligation/advice expressions.</p> <p>Allow bags in attribute assignment expressions.</p> <p>Use the new daytimeduration and yearmonthduration identifiers consistently.</p>
WD 13	14 Dec 2009	Erik Rissanen	<p>Fix small inconsistency in number of arguments to the multiply function.</p> <p>Generalize higher order bag functions.</p> <p>Add ContextSelectorId to attribute selector.</p> <p>Use &lt;Policy(Set)IdReference&gt; in &lt;PolicyIdList&gt;.</p> <p>Fix typos and formatting issues.</p> <p>Make the conformance section clearly reference the functional requirements in the spec.</p> <p>Conformance tests are no longer hosted by Sun.</p>
WD 14	17 Dec 2009	Erik Rissanen	Update acknowledgments
WD 15		Erik Rissanen	<p>Replace DecisionCombiningAlgorithm with a simple Boolean for CombinedDecision.</p> <p>Restrict &lt;Content&gt; to a single child element</p>



			and update the <AttributeSelector> and XPathExpression data type accordingly.
WD 16	12 Jan 2010	Erik Rissanen	Updated cross references Fix typos and minor inconsistencies. Simplify schema of <PolicyIdentifierList> Refactor some of the text to make it easier to understand. Update acknowledgments
WD 17	8 Mar 2010	Erik Rissanen	Updated cross references. Fixed OASIS style issues.
WD 18	23 Jun 2010	Erik Rissanen	Fixed typos in examples. Fixed typos in schema fragments.
WD 19	14 April 2011	Erik Rissanen	Updated function identifiers for new duration functions. Listed old identifiers as planned for deprecation. Added example for the X500Name-match function. Removed the (broken) Haskell definitions of the higher order functions. Clarified behavior of extended indeterminate in context of legacy combining algorithms or an Indeterminate target. Removed <Condition> from the expression substitution group. Specified argument order for subtract, divide and mod functions. Specified datatype to string conversion form to those functions which depend on it. Specified Indeterminate value for functions which convert strings to another datatype if the string is not a valid lexicographical representation of the datatype. Removed higher order functions for ip address and dns name.
WD 20	24 May 2011	Erik Rissanen	Fixed typo between "first" and "second" arguments in rfc822Name-match function. Removed duplicate word "string" in a couple of places. Improved and reorganized the text about extended indeterminate processing and Rule/Policy/PolicySet evaluation. Explicitly stated that an implementation is conformant regardless of its internal workings as long as the external result is the same as in this specification. Changed requirement on Indeterminate behavior at the top of section A.3 which

			conflicted with Boolean function definitions.
WD 21	28 Jun 2011	Erik Rissanen	<p>Redefined combining algorithms so they explicitly evaluate their children in the pseudocode.</p> <p>Changed wording in 7.12 and 7.13 to clarify that the combining algorithm applies to the children only, not the target.</p> <p>Removed wording in attribute category definitions about the attribute categories appearing multiple times since bags of bags are not supported,</p> <p>Fixed many small typos.</p> <p>Clarified wording about combiner parameters.</p>
WD 22	28 Jun 2011	Erik Rissanen	Fix typos in combining algorithm pseudo code.
WD 23	19 Mar 2012	Erik Rissanen	<p>Reformat references to OASIS specs.</p> <p>Define how XACML identifiers are matched.</p> <p>Do not highlight "actions" with the glossary term meaning in section 2.12.</p> <p>Fix minor typos.</p> <p>Make a reference to the full list of combining algorithms from the introduction.</p> <p>Clarified behavior of the context handler.</p> <p>Renamed higher order functions which were generalized in an earlier working draft.</p> <p>Add missing line in schema fragment for &lt;AttributeDesignator&gt;</p> <p>Removed reference to reuse of rules in section 2.2. There is no mechanism in XACML itself to re-use rules, though of course a tool could create copies as a form of "re-use".</p>

6129