



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

## OASIS Standard incorporating Public Review Draft 02 of Errata 01

11 May 2017

### Specification URIs

#### This version:

<http://docs.oasis-open.org/xacml/3.0/errata01/csprd02/xacml-3.0-core-spec-errata01-csprd02-complete.doc> (Authoritative)  
<http://docs.oasis-open.org/xacml/3.0/errata01/csprd02/xacml-3.0-core-spec-errata01-csprd02-complete.html>  
<http://docs.oasis-open.org/xacml/3.0/errata01/csprd02/xacml-3.0-core-spec-errata01-csprd02-complete.pdf>

#### Previous version:

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

#### 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@axiomantics.com](mailto:erik@axiomantics.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 02 / Public Review Draft 02. <http://docs.oasis-open.org/xacml/3.0/errata01/csprd02/xacml-3.0-core-spec-errata01-csprd02.html>.
- *eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01 (redlined)*. OASIS Standard change-marked (redlined) with Public Review Draft 02 of Errata 01: <http://docs.oasis-open.org/xacml/3.0/errata01/csprd02/xacml-3.0-core-spec-errata01-csprd02-redlined.html>.

- XML schema – unmodified from OASIS Standard: <http://docs.oasis-open.org/xacml/3.0/errata01/csprd02/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* incorporating the changes defined in Public Review Draft 02 of Errata 01.

**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/>.

This document is provided under the [RF on Limited Terms](#) Mode of the [OASIS IPR Policy](#), the mode chosen when the Technical Committee was established. 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-complete]**

*eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01*. Edited by Erik Rissanen. 11 May 2017. OASIS Standard incorporating Public Review Draft 02 of Errata 01.

<http://docs.oasis-open.org/xacml/3.0/errata01/csprd02/xacml-3.0-core-spec-errata01-csprd02-complete.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> .....	58
5.31 Element <AttributeValue> .....	59
5.32 Element <Obligations> .....	60
5.33 Element <AssociatedAdvice> .....	60
5.34 Element <Obligation> .....	60
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> .....	65
5.44 Element <Attributes> .....	66
5.45 Element <Content> .....	66
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>	69
5.52	Element <AttributesReference>	70
5.53	Element <Decision>	70
5.54	Element <Status>	70
5.55	Element <StatusCode>	71
5.56	Element <StatusMessage>	71
5.57	Element <StatusDetail>	71
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 Introduction

## 1.1 Glossary (non-normative)

### 1.1.1 Preferred terms

#### Access

Performing an *action*

#### Access control

Controlling *access* in accordance with a *policy* or *policy set*

#### Action

An operation on a *resource*

#### Advice

A supplementary piece of information in a *policy* or *policy set* which is provided to the *PEP* with the *decision* of the *PDP*.

#### Applicable policy

The set of *policies* and *policy sets* that governs *access* for a specific *decision request*

#### Attribute

Characteristic of a *subject*, *resource*, *action* or *environment* that may be referenced in a *predicate* or *target* (see also – *named attribute*)

#### Authorization decision

The result of evaluating *applicable policy*, returned by the *PDP* to the *PEP*. A function that evaluates to "Permit", "Deny", "Indeterminate" or "NotApplicable", and (optionally) a set of *obligations and advice*

#### Bag

An unordered collection of values, in which there may be duplicate values

#### Condition

An expression of *predicates*. A function that evaluates to "True", "False" or "Indeterminate"

#### Conjunctive sequence

A sequence of *predicates* combined using the logical 'AND' operation

#### Context

The canonical representation of a *decision request* and an *authorization decision*

#### Context handler

The system entity that converts *decision requests* in the native request format to the XACML canonical form, coordinates with Policy Information Points to add attribute values to the request context, and converts *authorization decisions* in the XACML canonical form to the native response format

#### Decision

The result of evaluating a *rule*, *policy* or *policy set*

#### Decision request

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

39	<b>Disjunctive sequence</b>
40	A sequence of <i>predicates</i> combined using the logical 'OR' operation
41	<b>Effect</b>
42	The intended consequence of a satisfied <i>rule</i> (either "Permit" or "Deny")
43	<b>Environment</b>
44	The set of <i>attributes</i> that are relevant to an <i>authorization decision</i> and are independent of a particular <i>subject, resource</i> or <i>action</i>
45	
46	<b>Identifier equality</b>
47	The identifier equality operation which is defined in section 7.20.
48	<b>Issuer</b>
49	A set of <i>attributes</i> describing the source of a <i>policy</i>
50	<b>Named attribute</b>
51	A specific instance of an <i>attribute</i> , determined by the <i>attribute</i> name and type, the identity of the
52	<i>attribute</i> holder (which may be of type: <i>subject, resource, action</i> or <i>environment</i> ) and
53	(optionally) the identity of the issuing authority
54	<b>Obligation</b>
55	An operation specified in a <i>rule, policy</i> or <i>policy set</i> that should be performed by the <i>PEP</i> in
56	conjunction with the enforcement of an <i>authorization decision</i>
57	<b>Policy</b>
58	A set of <i>rules</i> , an identifier for the <i>rule-combining algorithm</i> and (optionally) a set of
59	<i>obligations</i> or <i>advice</i> . May be a component of a <i>policy set</i>
60	<b>Policy administration point (PAP)</b>
61	The system entity that creates a <i>policy</i> or <i>policy set</i>
62	<b>Policy-combining algorithm</b>
63	The procedure for combining the <i>decision</i> and <i>obligations</i> from multiple <i>policies</i>
64	<b>Policy decision point (PDP)</b>
65	The system entity that evaluates <i>applicable policy</i> and renders an <i>authorization decision</i> .
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	<b>Policy enforcement point (PEP)</b>
70	The system entity that performs <i>access control</i> , by making <i>decision requests</i> and enforcing
71	<i>authorization decisions</i> . 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	<b>Policy information point (PIP)</b>
76	The system entity that acts as a source of <i>attribute</i> values
77	<b>Policy set</b>
78	A set of <i>policies</i> , other <i>policy sets</i> , a <i>policy-combining algorithm</i> and (optionally) a set of
79	<i>obligations</i> or <i>advice</i> . May be a component of another <i>policy set</i>
80	<b>Predicate</b>
81	A statement about <i>attributes</i> whose truth can be evaluated
82	<b>Resource</b>

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.

117

118 Listings of XACML schema appear like this.

119

120 Example code listings appear like this.

121

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].

- 127 • The prefix `xs:` stands for the W3C XML Schema namespace **[XS]**.
- 128 • The prefix `xf:` stands for the XQuery 1.0 and XPath 2.0 Function and Operators specification namespace **[XF]**.
- 129
- 130 • The prefix `xml:` stands for the XML namespace <http://www.w3.org/XML/1998/namespace>.
- 131 This specification uses the following typographical conventions in text: `<XACMLElement>`,
- 132 `<ns:ForeignElement>`, Attribute, Datatype, OtherCode. Terms in ***bold-face italic*** are intended
- 133 to have the meaning defined in the Glossary.

### 134 1.3 Schema organization and namespaces

135 The XACML syntax is defined in a schema associated with the following XML namespace:  
 136 `urn:oasis:names:tc:xacml:3.0:core:schema:wd-17`

### 137 1.4 Normative References

138	<b>[CMF]</b>	Martin J. Dürst et al, eds., <i>Character Model for the World Wide Web 1.0: Fundamentals</i> , W3C Recommendation 15 February 2005, <a href="http://www.w3.org/TR/2005/REC-charmod-20050215/">http://www.w3.org/TR/2005/REC-charmod-20050215/</a>
139		
140		
141	<b>[DS]</b>	D. Eastlake et al., <i>XML-Signature Syntax and Processing</i> , <a href="http://www.w3.org/TR/xmldsig-core/">http://www.w3.org/TR/xmldsig-core/</a> , World Wide Web Consortium.
142		
143	<b>[exc-c14n]</b>	J. Boyer et al, eds., <i>Exclusive XML Canonicalization, Version 1.0</i> , W3C Recommendation 18 July 2002, <a href="http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/">http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/</a>
144		
145		
146	<b>[Hancock]</b>	Hancock, <i>Polymorphic Type Checking</i> , in Simon L. Peyton Jones, <i>Implementation of Functional Programming Languages</i> , Section 8, Prentice-Hall International, 1987.
147		
148		
149	<b>[Hier]</b>	<i>XACML v3.0 Hierarchical Resource Profile Version 1.0</i> . 11 March 2010. Committee Specification Draft 03. <a href="http://docs.oasis-open.org/xacml/3.0/xacml-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</a>
150		
151		
152	<b>[IEEE754]</b>	IEEE Standard for Binary Floating-Point Arithmetic 1985, ISBN 1-5593-7653-8, IEEE Product No. SH10116-TBR.
153		
154	<b>[INFOSET]</b>	XML Information Set (Second Edition), W3C Recommendation 4 February 2004, available at <a href="https://www.w3.org/TR/xml-infoset/">https://www.w3.org/TR/xml-infoset/</a>
155		
156	<b>[ISO10181-3]</b>	ISO/IEC 10181-3:1996 Information technology – Open Systems Interconnection - Security frameworks for open systems: Access control framework.
157		
158	<b>[Kudo00]</b>	Kudo M and Hada S, <i>XML document security based on provisional authorization</i> , Proceedings of the Seventh ACM Conference on Computer and Communications Security, Nov 2000, Athens, Greece, pp 87-96.
159		
160		
161	<b>[LDAP-1]</b>	RFC2256, <i>A summary of the X500(96) User Schema for use with LDAPv3</i> , Section 5, M Wahl, December 1997, <a href="http://www.ietf.org/rfc/rfc2256.txt">http://www.ietf.org/rfc/rfc2256.txt</a>
162		
163	<b>[LDAP-2]</b>	RFC2798, <i>Definition of the inetOrgPerson</i> , M. Smith, April 2000 <a href="http://www.ietf.org/rfc/rfc2798.txt">http://www.ietf.org/rfc/rfc2798.txt</a>
164		
165	<b>[MathML]</b>	<i>Mathematical Markup Language (MathML)</i> , Version 2.0, W3C Recommendation, 21 October 2003. Available at: <a href="http://www.w3.org/TR/2003/REC-MathML2-20031021/">http://www.w3.org/TR/2003/REC-MathML2-20031021/</a>
166		
167		
168	<b>[Multi]</b>	OASIS Committee Draft 03, <i>XACML v3.0 Multiple Decision Profile Version 1.0</i> , 11 March 2010, <a href="http://docs.oasis-open.org/xacml/3.0/xacml-3.0-multiple-v1-spec-cd-03-en.doc">http://docs.oasis-open.org/xacml/3.0/xacml-3.0-multiple-v1-spec-cd-03-en.doc</a>
169		
170		
171	<b>[Perritt93]</b>	Perritt, H. Knowbots, <i>Permissions Headers and Contract Law</i> , Conference on Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment, April 1993. Available at: <a href="http://www.ifla.org/documents/infopol/copyright/perh2.txt">http://www.ifla.org/documents/infopol/copyright/perh2.txt</a>
172		
173		
174		

175	<b>[RBAC]</b>	David Ferraiolo and Richard Kuhn, <i>Role-Based Access Controls</i> , 15th National Computer Security Conference, 1992.
176		
177	<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.
178		
179	<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>
180		
181	<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>
182		
183	<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>
184		
185	<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>
186		
187	<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>
188		
189	<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>
190		
191		
192	<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>
193		
194		
195	<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>
196		
197		
198	<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>
199		
200		
201	<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>
202		
203		
204	<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>
205		
206		
207	<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>
208		
209	<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>
210		
211	<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>
212		
213		
214	<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>
215		

## 216 1.5 Non-Normative References

217	<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.
218		
219		
220	<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.
221		
222		
223	<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.
224		

225

## 2 Background (non-normative)

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

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

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

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

### 2.1 Requirements

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

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



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

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

## 276 2.2 Rule and policy combining

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

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

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

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

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

## 295 2.3 Combining algorithms

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

- 303 • Deny-overrides (Ordered and Unordered),
- 304 • Permit-overrides (Ordered and Unordered),
- 305 • First-applicable and
- 306 • Only-one-applicable.

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

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

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

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

319 applicable, the result of the combining algorithm is the result of evaluating the single **applicable policy** or  
320 **policy set**.

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

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

## 324 2.4 Multiple subjects

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

## 331 2.5 Policies based on subject and resource attributes

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

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

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

## 348 2.6 Multi-valued attributes

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

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

## 358 2.7 Policies based on resource contents

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

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



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

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

## 369 2.8 Operators

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

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

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

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

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

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

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

## 398 2.9 Policy distribution

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

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

## 410 2.10 Policy indexing

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

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

415 Two approaches are supported:

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

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

## 424 2.11 Abstraction layer

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

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

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

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

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

## 446 2.12 Actions performed in conjunction with enforcement

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

## 456 2.13 Supplemental information about a decision

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

### 3 Models (non-normative)

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

#### 3.1 Data-flow model

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

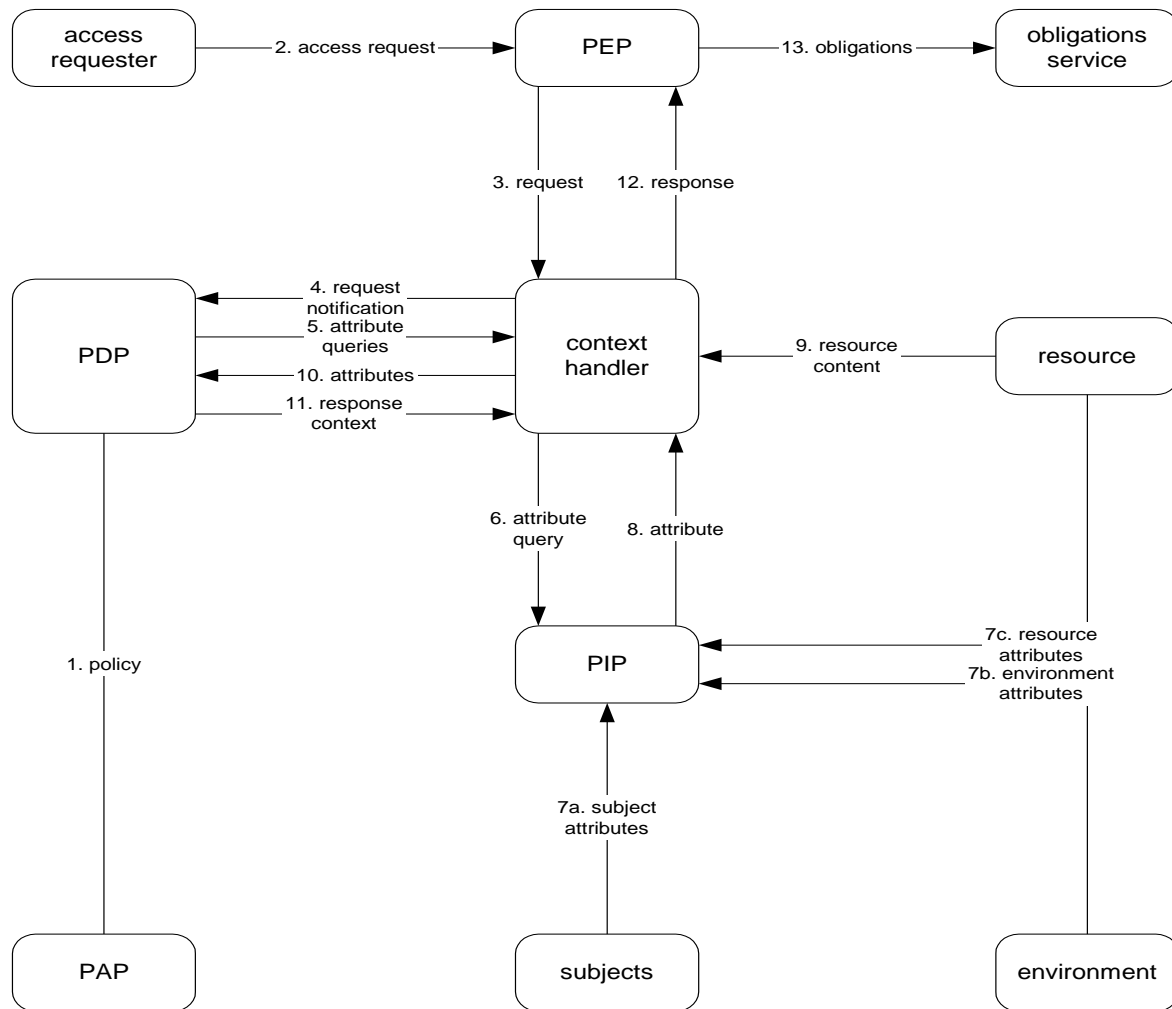


Figure 1 - Data-flow diagram

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

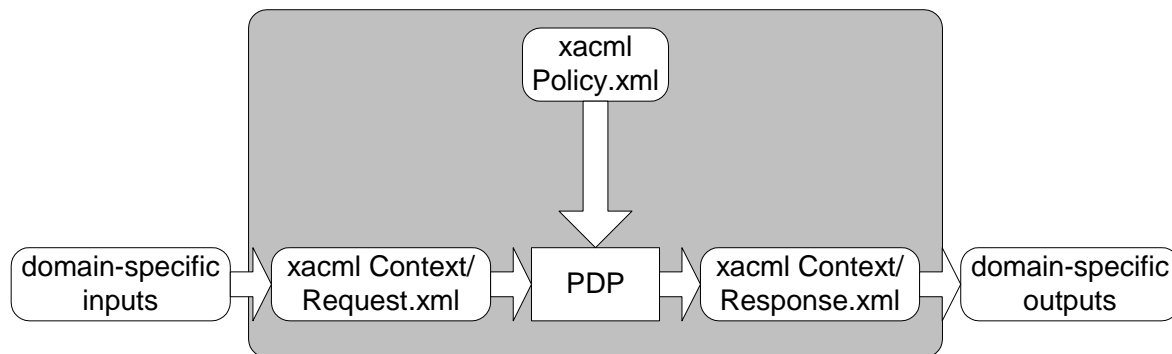
The model operates by the following steps.

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

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

### 496 3.2 XACML context

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



508  
509 *Figure 2 - XACML context*

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

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

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

515 **3.3 Policy language model**

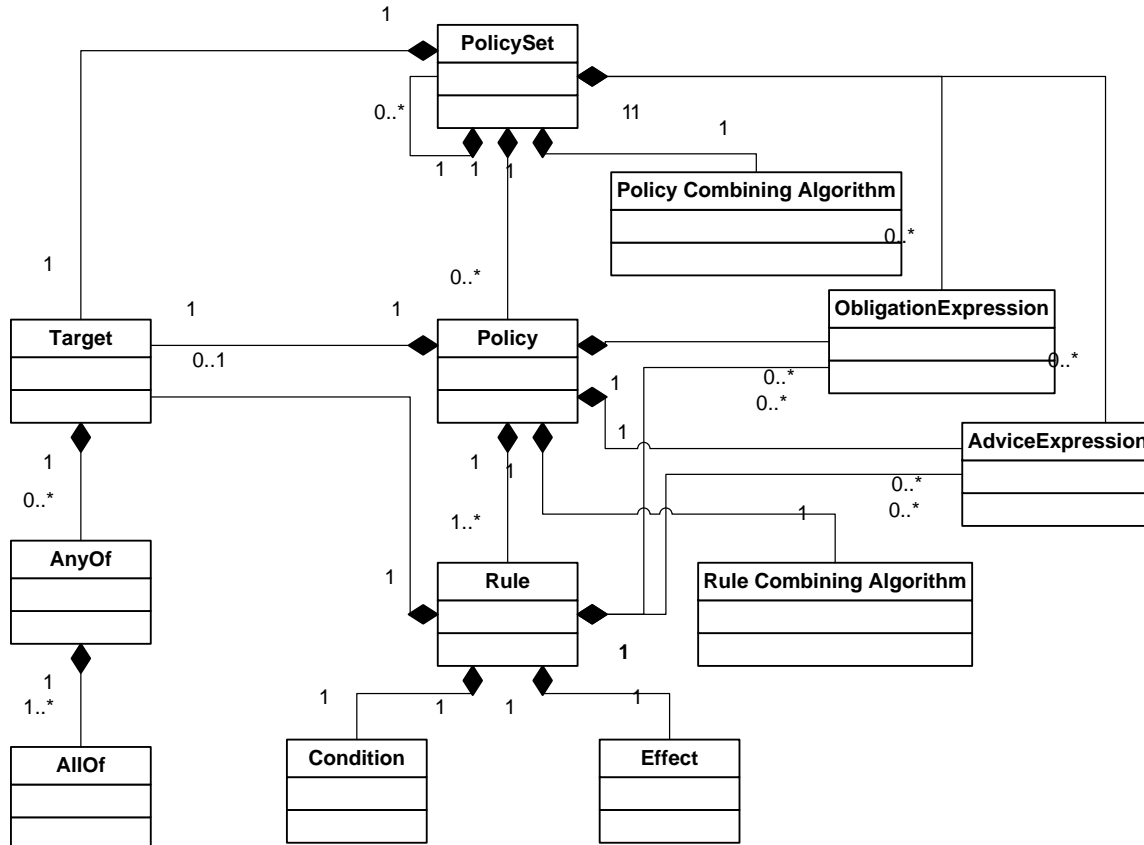
516 *The policy language model is shown in*

517 Figure 3. The main components of the model are:

- 518 • **Rule**;
- 519 • **Policy**, and
- 520 • **Policy set**.

521 These are described in the following sub-sections.

522



523

524

525 *Figure 3 - Policy language model*

526 **3.3.1 Rule**

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

- 530 • a **target**,
- 531 • an **effect**,
- 532 • a **condition**,
- 533 • **obligation** expressions, and
- 534 • **advice** expressions

535 These are discussed in the following sub-sections.

### 536 3.3.1.1 Rule target

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

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

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

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

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

557 In the case of **subjects**, there is no real entity that corresponds to such a node. So, names of this type  
558 always refer to the set of **subjects** subordinate in the name structure to the identified node.  
559 Consequently, non-leaf **subject** names should not be used in equality functions, only in match functions,  
560 such as "urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match" not  
561 "urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal" (see Appendix 10.2.9).

### 562 3.3.1.2 Effect

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

### 565 3.3.1.3 Condition

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

### 568 3.3.1.4 Obligation expressions

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

570 When a **PDP** evaluates a **rule** containing **obligation** expressions, it evaluates the **obligation** expressions  
571 into **obligations** and returns certain of those **obligations** to the **PEP** in the response **context**. Section  
572 7.18 explains which **obligations** are to be returned.

### 573 3.3.1.5 Advice

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

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

## 578 3.3.2 Policy

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



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

### 587 3.3.2.1 Policy target

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

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

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

### 607 3.3.2.2 Rule-combining algorithm

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

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

### 613 3.3.2.3 Obligation expressions

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

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

### 618 3.3.2.4 Advice

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

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

### 623 3.3.3 Policy set

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

- 625 • a **target**;
- 626 • a **policy-combining algorithm**-identifier
- 627 • a set of **policies**;
- 628 • **obligation** expressions, and
- 629 • **advice** expressions

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

#### 632 3.3.3.1 Policy-combining algorithm

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

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

#### 639 3.3.3.2 Obligation expressions

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

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

#### 645 3.3.3.3 Advice expressions

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

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



651

## 4 Examples (non-normative)

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

656

### 4.1 Example one

657

#### 4.1.1 Example policy

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

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

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

```
664 [a1] <?xml version="1.0" encoding="UTF-8"?>
665 [a2] <Policy
666 [a3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
667 [a4]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
668 [a5]   xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
669 [a6]   http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
670 [a7]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:SimplePolicy1"
671 [a8]   Version="1.0"
672 [a9]   RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">
673 [a10]  <Description>
674 [a11]    Medi Corp access control policy
675 [a12]  </Description>
676 [a13]  <Target/>
677 [a14]  <Rule
678 [a15]    RuleId="urn:oasis:names:tc:xacml:3.0:example:SimpleRule1"
679 [a16]    Effect="Permit">
680 [a17]    <Description>
681 [a18]      Any subject with an e-mail name in the med.example.com domain
682 [a19]      can perform any action on any resource.
683 [a20]    </Description>
684 [a21]    <Target>
685 [a22]      <AnyOf>
686 [a23]        <AllOf>
687 [a24]          <Match
688 [a25]            MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
689 [a26]              <AttributeValue
690 [a27]                DataType="http://www.w3.org/2001/XMLSchema#string"
691 [a28]                >med.example.com</AttributeValue>
692 [a29]              <AttributeDesignator
693 [a30]                MustBePresent="false"
694 [a31]                Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
695 [a32] subject"
696 [a33]                AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
697 [a34]                DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
698 [a35]              </Match>
699 [a36]            </AllOf>
700 [a37]          </AnyOf>
701 [a38]        </Target>
702 [a39]      </Rule>
703 [a40]    </Policy>
```

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

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

707 [a3] - [a4] are XML namespace declarations.

708 [a3] gives a URN for the XACML **policies** schema.

709 [a7] assigns a name to this **policy** instance. The name of a **policy** has to be unique for a given **PDP** so  
710 that there is no ambiguity if one **policy** is referenced from another **policy**. The version attribute specifies  
711 the version of this policy is "1.0".

712 [a9] specifies the algorithm that will be used to resolve the results of the various **rules** that may be in the  
713 **policy**. The deny-overrides **rule-combining algorithm** specified here says that, if any **rule** evaluates to  
714 "Deny", then the **policy** must return "Deny". If all **rules** evaluate to "Permit", then the **policy** must return  
715 "Permit". The **rule-combining algorithm**, which is fully described in Appendix Appendix C, also says  
716 what to do if an error were to occur when evaluating any **rule**, and what to do with **rules** that do not apply  
717 to a particular **decision request**.

718 [a10] - [a12] provide a text description of the **policy**. This description is optional.

719 [a13] describes the **decision requests** to which this **policy** applies. If the **attributes** in a **decision**  
720 **request** do not match the values specified in the **policy target**, then the remainder of the **policy** does not  
721 need to be evaluated. This **target** section is useful for creating an index to a set of **policies**. In this  
722 simple example, the **target** section says the **policy** is applicable to any **decision request**.

723 [a14] introduces the one and only **rule** in this simple **policy**.

724 [a15] specifies the identifier for this **rule**. Just as for a **policy**, each **rule** must have a unique identifier (at  
725 least unique for any **PDP** that will be using the **policy**).

726 [a16] says what **effect** this **rule** has if the **rule** evaluates to "True". **Rules** can have an **effect** of either  
727 "Permit" or "Deny". In this case, if the **rule** is satisfied, it will evaluate to "Permit", meaning that, as far as  
728 this one **rule** is concerned, the requested **access** should be permitted. If a **rule** evaluates to "False",  
729 then it returns a result of "NotApplicable". If an error occurs when evaluating the **rule**, then the **rule**  
730 returns a result of "Indeterminate". As mentioned above, the **rule-combining algorithm** for the **policy**  
731 specifies how various **rule** values are combined into a single **policy** value.

732 [a17] - [a20] provide a text description of this **rule**. This description is optional.

733 [a21] introduces the **target** of the **rule**. As described above for the **target** of a **policy**, the **target** of a **rule**  
734 describes the **decision requests** to which this **rule** applies. If the **attributes** in a **decision request** do  
735 not match the values specified in the **rule target**, then the remainder of the **rule** does not need to be  
736 evaluated, and a value of "NotApplicable" is returned to the **rule** evaluation.

737 The **rule target** is similar to the **target** of the **policy** itself, but with one important difference. [a22] - [a36]  
738 spells out a specific value that the **subject** in the **decision request** must match. The <Match> element  
739 specifies a matching function in the MatchId attribute, a literal value of "med.example.com" and a pointer  
740 to a specific **subject attribute** in the request **context** by means of the <AttributeDesignator>  
741 element with an **attribute** category which specifies the **access subject**. The matching function will be  
742 used to compare the literal value with the value of the **subject attribute**. Only if the match returns "True"  
743 will this **rule** apply to a particular **decision request**. If the match returns "False", then this **rule** will return  
744 a value of "NotApplicable".

745 [a38] closes the **rule**. In this **rule**, all the work is done in the <Target> element. In more complex **rules**,  
746 the <Target> may have been followed by a <Condition> element (which could also be a set of  
747 **conditions** to be ANDed or ORed together).

748 [a39] closes the **policy**. As mentioned above, this **policy** has only one **rule**, but more complex **policies**  
749 may have any number of **rules**.

## 750 4.1.2 Example request context

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

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

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

```

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

```

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

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

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

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

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

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

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

### 814 4.1.3 Example response context

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

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

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

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

832 [c6] closes the response *context*.

## 833 4.2 Example two

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

### 837 4.2.1 Example medical record instance

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

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

```

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

```

## 922 4.2.2 Example request context

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

```

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

```

```

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

```

1001 [e2] - [e4] Standard namespace declarations.

1002 [e6] - [e27] **Access subject attributes** are placed in the urn:oasis:names:tc:xacml:1.0:subject-  
1003 category:access-subject **attribute** category of the <Request> element. Each **attribute** consists of the  
1004 **attribute** meta-data and the **attribute** value. There is only one **subject** involved in this request. This  
1005 value of the **attribute** category denotes the identity for which the request was issued.

1006 [e8] - [e13] **Subject** subject-id **attribute**.

1007 [e14] - [e20] **Subject** role **attribute**.

1008 [e21] - [e26] **Subject** physician-id **attribute**.



1009 [e28] - [e56] **Resource attributes** are placed in the urn:oasis:names:tc:xacml:3.0:attribute-  
 1010 category:resource **attribute** category of the <Request> element. Each **attribute** consists of **attribute**  
 1011 meta-data and an **attribute** value.

1012 [e30] - [e42] **Resource** content. The XML **resource** instance, **access** to all or part of which may be  
 1013 requested, is placed here.

1014 [e43] - [e49] The identifier of the **Resource** instance for which **access** is requested, which is an XPath  
 1015 expression into the <Content> element that selects the data to be accessed.

1016 [e57] - [e64] **Action attributes** are placed in the urn:oasis:names:tc:xacml:3.0:attribute-category:action  
 1017 **attribute** category of the <Request> element.

1018 [e59] - [e63] **Action** identifier.

### 1019 4.2.3 Example plain-language rules

1020 The following plain-language **rules** are to be enforced:

- 1021 Rule 1: A person, identified by his or her patient number, may read any record for which he or she is  
 1022 the designated patient.
- 1023 Rule 2: A person may read any record for which he or she is the designated parent or guardian, and  
 1024 for which the patient is under 16 years of age.
- 1025 Rule 3: A physician may write to any medical element for which he or she is the designated primary  
 1026 care physician, provided an email is sent to the patient.
- 1027 Rule 4: An administrator shall not be permitted to read or write to medical elements of a patient  
 1028 record.

1029 These **rules** may be written by different **PAPs** operating independently, or by a single **PAP**.

### 1030 4.2.4 Example XACML rule instances

#### 1031 4.2.4.1 Rule 1

1032 **Rule 1** illustrates a simple **rule** with a single <Condition> element. It also illustrates the use of the  
 1033 <VariableDefinition> element to define a function that may be used throughout the **policy**. The  
 1034 following XACML <Rule> instance expresses **Rule 1**:

```

1035 [f1] <?xml version="1.0" encoding="UTF-8"?>
1036 [f2] <Policy
1037 [f3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1038 [f4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1039 [f5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1040 [f6]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1041 [f7]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:1"
1042 [f8]   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1043   algorithm:deny-overrides"
1044 [f9]   Version="1.0">
1045 [f10]   <PolicyDefaults>
1046 [f11]     <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1047 [f12]   </PolicyDefaults>
1048 [f13]   <Target/>
1049 [f14]   <VariableDefinition VariableId="17590034">
1050 [f15]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1051 [f16]       <Apply
1052 [f17]         FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1053 [f18]           <AttributeDesignator
1054 [f19]             MustBePresent="false"
1055 [f20]             Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
1056   subject"
1057 [f21]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:patient-
1058   number"
1059 [f22]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1060 [f23]         </Apply>
1061 [f24]       <Apply
1062 [f25]         FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
  
```

```

1063 [f26]         <AttributeSelector
1064 [f27]             MustBePresent="false"
1065 [f28]             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1066 [f29]             Path="md:record/md:patient/md:patient-number/text()"
1067 [f30]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1068 [f31]         </Apply>
1069 [f32]     </Apply>
1070 [f33] </VariableDefinition>
1071 [f34] <Rule
1072 [f35]     RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"
1073 [f36]     Effect="Permit">
1074 [f37]     <Description>
1075 [f38]         A person may read any medical record in the
1076 [f39]         http://www.med.example.com/schemas/record.xsd namespace
1077 [f40]         for which he or she is the designated patient
1078 [f41]     </Description>
1079 [f42]     <Target>
1080 [f43]         <AnyOf>
1081 [f44]             <AllOf>
1082 [f45]                 <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1083 [f46]                     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1084 [f47]                         >urn:example:med:schemas:record</AttributeValue>
1085 [f48]                     <AttributeDesignator
1086 [f49]                         MustBePresent="false"
1087 [f50]                         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1088 [f51]                         AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1089 [f52]                         DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1090 [f53]                 </Match>
1091 [f54]                 <Match
1092 [f55]                     MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1093 [f56]                         <AttributeValue
1094 [f57]                             DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1095 [f58]                             XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1096 [f59]                             >md:record</AttributeValue>
1097 [f60]                         <AttributeDesignator
1098 [f61]                             MustBePresent="false"
1099 [f62]                             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1100 [f63]                             AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1101 [f64]                             DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1102 [f65]                         </Match>
1103 [f66]                     </AllOf>
1104 [f67]                 </AnyOf>
1105 [f68]             <AnyOf>
1106 [f69]                 <AllOf>
1107 [f70]                     <Match
1108 [f71]                         MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1109 [f72]                             <AttributeValue
1110 [f73]                                 DataType="http://www.w3.org/2001/XMLSchema#string"
1111 [f74]                                 >read</AttributeValue>
1112 [f75]                             <AttributeDesignator
1113 [f76]                                 MustBePresent="false"
1114 [f77]                                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1115 [f78]                                 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1116 [f79]                                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1117 [f80]                             </Match>
1118 [f81]                     </AllOf>
1119 [f82]                 </AnyOf>
1120 [f83]             </Target>
1121 [f84]         <Condition>
1122 [f85]             <VariableReference VariableId="17590034"/>
1123 [f86]         </Condition>
1124 [f87]     </Rule>
1125 [f88] </Policy>

```

1126 [f3] - [f6] XML namespace declarations.

1127 [f11] XPath expressions in the **policy** are to be interpreted according to the 1.0 version of the XPath specification.

1129 [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**.



1131 [f15] The `FunctionId` attribute names the function to be used for comparison. In this case, comparison  
1132 is done with the “urn:oasis:names:tc:xacml:1.0:function:string-equal” function; this function takes two  
1133 arguments of type “http://www.w3.org/2001/XMLSchema#string”.

1134 [f17] The first argument of the variable definition is a function specified by the `FunctionId` attribute.  
1135 Since urn:oasis:names:tc:xacml:1.0:function:string-equal takes arguments of type  
1136 “http://www.w3.org/2001/XMLSchema#string” and `AttributeDesignator` selects a **bag** of type  
1137 “http://www.w3.org/2001/XMLSchema#string”, “urn:oasis:names:tc:xacml:1.0:function:string-one-and-  
1138 only” is used. This function guarantees that its argument evaluates to a **bag** containing exactly one  
1139 value.

1140 [f18] The `AttributeDesignator` selects a **bag** of values for the patient-number **subject attribute** in  
1141 the request **context**.

1142 [f25] The second argument of the variable definition is a function specified by the `FunctionId` attribute.  
1143 Since “urn:oasis:names:tc:xacml:1.0:function:string-equal” takes arguments of type  
1144 “http://www.w3.org/2001/XMLSchema#string” and the `AttributeSelector` selects a **bag** of type  
1145 “http://www.w3.org/2001/XMLSchema#string”, “urn:oasis:names:tc:xacml:1.0:function:string-one-and-  
1146 only” is used. This function guarantees that its argument evaluates to a **bag** containing exactly one  
1147 value.

1148 [f26] The `<AttributeSelector>` element selects a **bag** of values from the **resource** content using a  
1149 free-form XPath expression. In this case, it selects the value of the patient-number in the **resource**.  
1150 Note that the namespace prefixes in the XPath expression are resolved with the standard XML  
1151 namespace declarations.

1152 [f35] **Rule** identifier.

1153 [f36] **Rule effect** declaration. When a **rule** evaluates to ‘True’ it emits the value of the `Effect` attribute.  
1154 This value is then combined with the `Effect` values of other **rules** according to the **rule-combining**  
1155 **algorithm**.

1156 [f37] - [f41] Free form description of the **rule**.

1157 [f42] - [f83] A **rule target** defines a set of **decision requests** that the **rule** is intended to evaluate.

1158 [f43] - [f67] The `<AnyOf>` element contains a **disjunctive sequence** of `<AllOf>` elements. In this  
1159 example, there is just one.

1160 [f44] - [f66] The `<AllOf>` element encloses the **conjunctive sequence** of `Match` elements. In this  
1161 example, there are two.

1162 [f45] - [f53] The first `<Match>` element compares its first and second child elements according to the  
1163 matching function. A match is positive if the value of the first argument matches any of the values  
1164 selected by the second argument. This match compares the **target** namespace of the requested  
1165 document with the value of “urn:example:med:schemas:record”.

1166 [f45] The `MatchId` attribute names the matching function.

1167 [f46] - [f47] Literal **attribute** value to match.

1168 [f48] - [f52] The `<AttributeDesignator>` element selects the **target** namespace from the **resource**  
1169 contained in the request **context**. The **attribute** name is specified by the `AttributeId`.

1170 [f54] - [f65] The second `<Match>` element. This match compares the results of two XPath expressions  
1171 applied to the `<Content>` element of the **resource** category. The second XPath expression is the  
1172 location path to the requested XML element and the first XPath expression is the literal value “md:record”.  
1173 The “xpath-node-match” function evaluates to “True” if the requested XML element is below the  
1174 “md:record” element.

1175 [f68] - [f82] The `<AnyOf>` element contains a **disjunctive sequence** of `<AllOf>` elements. In this case,  
1176 there is just one `<AllOf>` element.

1177 [f69] - [f81] The `<AllOf>` element contains a **conjunctive sequence** of `<Match>` elements. In this case,  
1178 there is just one `<Match>` element.

1179 [f70] - [f80] The <Match> element compares its first and second child elements according to the matching  
1180 function. The match is positive if the value of the first argument matches any of the values selected by  
1181 the second argument. In this case, the value of the action-id **action attribute** in the request **context** is  
1182 compared with the literal value "read".

1183 [f84] - [f86] The <Condition> element. A **condition** must evaluate to "True" for the **rule** to be  
1184 applicable. This **condition** contains a reference to a variable definition defined elsewhere in the **policy**.

#### 1185 4.2.4.2 Rule 2

1186 **Rule 2** illustrates the use of a mathematical function, i.e. the <Apply> element with functionId  
1187 "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration" to calculate the date of the patient's  
1188 sixteenth birthday. It also illustrates the use of **predicate** expressions, with the functionId  
1189 "urn:oasis:names:tc:xacml:1.0:function:and". This example has one function embedded in the  
1190 <Condition> element and another one referenced in a <VariableDefinition> element.

```
1191 [g1] <?xml version="1.0" encoding="UTF-8"?>
1192 [g2] <Policy
1193 [g3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1194 [g4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1195 [g5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1196 [g6]   xmlns:xf="http://www.w3.org/2005/xpath-functions"
1197 [g7]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1198 [g8]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"
1199 [g9]   Version="1.0"
1200 [g10]  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1201      algorithm:deny-overrides">
1202 [g11]  <PolicyDefaults>
1203 [g12]    <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1204 [g13]  </PolicyDefaults>
1205 [g14]  <Target/>
1206 [g15]  <VariableDefinition VariableId="17590035">
1207 [g16]    <Apply
1208 [g17]      FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal">
1209 [g18]      <Apply
1210 [g19]        FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
1211 [g20]          <AttributeDesignator
1212 [g21]            MustBePresent="false"
1213 [g22]            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
1214 [g23]            AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"
1215 [g24]            DataType="http://www.w3.org/2001/XMLSchema#date"/>
1216 [g25]          </Apply>
1217 [g26]        <Apply
1218 [g27]          FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration">
1219 [g28]            <Apply
1220 [g29]              FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
1221 [g30]                <AttributeSelector
1222 [g31]                  MustBePresent="false"
1223 [g32]                  Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1224 [g33]                  Path="md:record/md:patient/md:patientDoB/text()"
1225 [g34]                  DataType="http://www.w3.org/2001/XMLSchema#date"/>
1226 [g35]                </Apply>
1227 [g36]              <AttributeValue
1228 [g37]                DataType="http://www.w3.org/2001/XMLSchema#yearMonthDuration"
1229 [g38]                >P16Y</AttributeValue>
1230 [g39]            </Apply>
1231 [g40]          </Apply>
1232 [g41]        </VariableDefinition>
1233 [g42]    <Rule
1234 [g43]      RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"
1235 [g44]      Effect="Permit">
1236 [g45]      <Description>
1237 [g46]        A person may read any medical record in the
1238 [g47]        http://www.med.example.com/records.xsd namespace
1239 [g48]        for which he or she is the designated parent or guardian,
1240 [g49]        and for which the patient is under 16 years of age
1241 [g50]      </Description>
1242 [g51]      <Target>
1243 [g52]        <AnyOf>
1244 [g53]          <AllOf>
```

```

1245 [g54] <Match
1246 [g55] MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1247 [g56] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1248 [g57] >urn:example:med:schemas:record</AttributeValue>
1249 [g58] <AttributeDesignator
1250 [g59] MustBePresent="false"
1251 [g60] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1252 [g61] AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1253 [g62] DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1254 [g63] </Match>
1255 [g64] <Match
1256 [g65] MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1257 [g66] <AttributeValue
1258 [g67] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1259 [g68] XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1260 [g69] >md:record</AttributeValue>
1261 [g70] <AttributeDesignator
1262 [g71] MustBePresent="false"
1263 [g72] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1264 [g73] AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1265 [g74] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1266 [g75] </Match>
1267 [g76] </AllOf>
1268 [g77] </AnyOf>
1269 [g78] <AnyOf>
1270 [g79] <AllOf>
1271 [g80] <Match
1272 [g81] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1273 [g82] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1274 [g83] >read</AttributeValue>
1275 [g84] <AttributeDesignator
1276 [g85] MustBePresent="false"
1277 [g86] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1278 [g87] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1279 [g88] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1280 [g89] </Match>
1281 [g90] </AllOf>
1282 [g91] </AnyOf>
1283 [g92] </Target>
1284 [g93] <Condition>
1285 [g94] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
1286 [g95] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1287 [g96] <Apply
1288 [g97] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1289 [g98] <AttributeDesignator
1290 [g99] MustBePresent="false"
1291 [g100] Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1292 [g101] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:parent-
guardian-id"
1293 [g102] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1294 [g103] </Apply>
1295 [g104] <Apply
1296 [g105] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1297 [g106] <AttributeSelector
1298 [g107] MustBePresent="false"
1299 [g108] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1300 [g109] Path="md:record/md:parentGuardian/md:parentGuardianId/text()"
1301 [g110] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1302 [g111] </Apply>
1303 [g112] </Apply>
1304 [g113] <VariableReference VariableId="17590035"/>
1305 [g114] </Apply>
1306 [g115] </Condition>
1307 [g116] </Rule>
1308 [g117] </Policy>

```

1310 [g15] - [g41] The <VariableDefinition> element contains part of the **condition** (i.e. is the patient  
1311 under 16 years of age?). The patient is under 16 years of age if the current date is less than the date  
1312 computed by adding 16 to the patient's date of birth.

1313 [g16] - [g40] "urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal" is used to compare the two date  
1314 arguments.

1315 [g18] - [g25] The first date argument uses “urn:oasis:names:tc:xacml:1.0:function:date-one-and-only” to  
 1316 ensure that the **bag** of values selected by its argument contains exactly one value of type  
 1317 “http://www.w3.org/2001/XMLSchema#date”.

1318 [g20] The current date is evaluated by selecting the “urn:oasis:names:tc:xacml:1.0:environment:current-  
 1319 date” **environment attribute**.

1320 [g26] - [g39] The second date argument uses “urn:oasis:names:tc:xacml:1.0:function:date-add-  
 1321 yearMonthDuration” to compute the date of the patient’s sixteenth birthday by adding 16 years to the  
 1322 patient’s date of birth. The first of its arguments is of type “http://www.w3.org/2001/XMLSchema#date”  
 1323 and the second is of type “http://www.w3.org/TR/2007/REC-xpath-functions-20070123/#dt-  
 1324 yearMonthDuration”.

1325 [g30] The <AttributeSelector> element selects the patient’s date of birth by taking the XPath  
 1326 expression over the **resource** content.

1327 [g36] - [g38] Year Month Duration of 16 years.

1328 [g51] - [g92] **Rule** declaration and **rule target**. See **Rule** 1 in Section 4.2.4.1 for the detailed explanation  
 1329 of these elements.

1330 [g93] - [g115] The <Condition> element. The **condition** must evaluate to “True” for the **rule** to be  
 1331 applicable. This **condition** evaluates the truth of the statement: the requestor is the designated parent or  
 1332 guardian and the patient is under 16 years of age. It contains one embedded <Apply> element and one  
 1333 referenced <VariableDefinition> element.

1334 [g94] The **condition** uses the “urn:oasis:names:tc:xacml:1.0:function:and” function. This is a Boolean  
 1335 function that takes one or more Boolean arguments (2 in this case) and performs the logical “AND”  
 1336 operation to compute the truth value of the expression.

1337 [g95] - [g112] The first part of the **condition** is evaluated (i.e. is the requestor the designated parent or  
 1338 guardian?). The function is “urn:oasis:names:tc:xacml:1.0:function:string-equal” and it takes two  
 1339 arguments of type “http://www.w3.org/2001/XMLSchema#string”.

1340 [g96] designates the first argument. Since “urn:oasis:names:tc:xacml:1.0:function:string-equal” takes  
 1341 arguments of type “http://www.w3.org/2001/XMLSchema#string”,  
 1342 “urn:oasis:names:tc:xacml:1.0:function:string-one-and-only” is used to ensure that the **subject attribute**  
 1343 “urn:oasis:names:tc:xacml:3.0:example:attribute:parent-guardian-id” in the request **context** contains  
 1344 exactly one value.

1345 [g98] designates the first argument. The value of the **subject attribute**  
 1346 “urn:oasis:names:tc:xacml:3.0:example:attribute:parent-guardian-id” is selected from the request **context**  
 1347 using the <AttributeDesignator> element.

1348 [g104] As above, the “urn:oasis:names:tc:xacml:1.0:function:string-one-and-only” is used to ensure that  
 1349 the **bag** of values selected by its argument contains exactly one value of type  
 1350 “http://www.w3.org/2001/XMLSchema#string”.

1351 [g106] The second argument selects the value of the <md:parentGuardianId> element from the  
 1352 **resource** content using the <AttributeSelector> element. This element contains a free-form XPath  
 1353 expression, pointing into the <Content> element of the resource category. Note that all namespace  
 1354 prefixes in the XPath expression are resolved with standard namespace declarations. The  
 1355 AttributeSelector evaluates to the **bag** of values of type  
 1356 “http://www.w3.org/2001/XMLSchema#string”.

1357 [g113] references the <VariableDefinition> element, where the second part of the **condition** is  
 1358 defined.

### 1359 4.2.4.3 Rule 3

1360 **Rule 3** illustrates the use of an **obligation** expression.

```

1361 [h1] <?xml version="1.0" encoding="UTF-8"?>
1362 [h2] <Policy
1363 [h3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1364 [h4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1365 [h5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

```

1366 [h6]      xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
1367 http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
1368 [h7]      xmlns:md="http://www.med.example.com/schemas/record.xsd"
1369 [h8]      PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:3"
1370 [h9]      Version="1.0"
1371 [h10]     RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1372 algorithm:deny-overrides">
1373 [h11]     <Description>
1374 [h12]       Policy for any medical record in the
1375 [h13]       http://www.med.example.com/schemas/record.xsd namespace
1376 [h14]     </Description>
1377 [h15]     <PolicyDefaults>
1378 [h16]       <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1379 [h17]     </PolicyDefaults>
1380 [h18]     <Target>
1381 [h19]       <AnyOf>
1382 [h20]         <AllOf>
1383 [h21]           <Match
1384 [h22]             MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1385 [h23]               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1386 [h24]                 >urn:example:med:schemas:record</AttributeValue>
1387 [h25]               <AttributeDesignator
1388 [h26]                 MustBePresent="false"
1389 [h27]                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1390 [h28]                 AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1391 [h29]                 DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1392 [h30]             </Match>
1393 [h31]           </AllOf>
1394 [h32]         </AnyOf>
1395 [h33]     </Target>
1396 [h34]     <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:3"
1397 [h35]       Effect="Permit">
1398 [h36]       <Description>
1399 [h37]         A physician may write any medical element in a record
1400 [h38]         for which he or she is the designated primary care
1401 [h39]         physician, provided an email is sent to the patient
1402 [h40]       </Description>
1403 [h41]       <Target>
1404 [h42]         <AnyOf>
1405 [h43]           <AllOf>
1406 [h44]             <Match
1407 [h45]               MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1408 [h46]                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1409 [h47]                   >physician</AttributeValue>
1410 [h48]                 <AttributeDesignator
1411 [h49]                   MustBePresent="false"
1412 [h50]                   Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1413 [h51]                   AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
1414 [h52]                   DataType="http://www.w3.org/2001/XMLSchema#string"/>
1415 [h53]                 </Match>
1416 [h54]             </AllOf>
1417 [h55]           </AnyOf>
1418 [h56]         <AnyOf>
1419 [h57]           <AllOf>
1420 [h58]             <Match
1421 [h59]               MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1422 [h60]                 <AttributeValue
1423 [h61]                   DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1424 [h62]                   XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1425 [h63]                   >md:record/md:medical</AttributeValue>
1426 [h64]                 <AttributeDesignator
1427 [h65]                   MustBePresent="false"
1428 [h66]                   Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1429 [h67]                   AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1430 [h68]                   DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1431 [h69]                 </Match>
1432 [h70]             </AllOf>
1433 [h71]           </AnyOf>
1434 [h72]         <AnyOf>
1435 [h73]           <AllOf>
1436 [h74]             <Match
1437 [h75]               MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1438 [h76]                 <AttributeValue

```

```

1439 [h77]         DataType="http://www.w3.org/2001/XMLSchema#string"
1440 [h78]         >write</AttributeValue>
1441 [h79]         <AttributeDesignator
1442 [h80]             MustBePresent="false"
1443 [h81]             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1444 [h82]             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1445 [h83]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1446 [h84]         </Match>
1447 [h85]     </AllOf>
1448 [h86] </AnyOf>
1449 [h87] </Target>
1450 [h88] <Condition>
1451 [h89]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1452 [h90]         <Apply
1453 [h91]             FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1454 [h92]             <AttributeDesignator
1455 [h93]                 MustBePresent="false"
1456 [h94]                 Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1457 [h95]                 AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:physician-id"
1458 [h96]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1459 [h97]             </Apply>
1460 [h98]         <Apply
1461 [h99]             FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1462 [h100]             <AttributeSelector
1463 [h101]                 MustBePresent="false"
1464 [h102]                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1465 [h103]                 Path="md:record/md:primaryCarePhysician/md:registrationID/text()"
1466 [h104]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1467 [h105]             </Apply>
1468 [h106]         </Apply>
1469 [h107]     </Condition>
1470 [h108] </Rule>
1471 [h109] <ObligationExpressions>
1472 [h110]     <ObligationExpression
1473 [h111]         ObligationId="urn:oasis:names:tc:xacml:example:obligation:email"
1474 [h112]         FulfillOn="Permit">
1475 [h113]         <AttributeAssignmentExpression
1476 [h114]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:mailto">
1477 [h115]             <AttributeSelector
1478 [h116]                 MustBePresent="true"
1479 [h117]                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1480 [h118]                 Path="md:record/md:patient/md:patientContact/md:email"
1481 [h119]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1482 [h120]             </AttributeAssignmentExpression>
1483 [h121]         <AttributeAssignmentExpression
1484 [h122]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
1485 [h123]             <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1486 [h124]                 >Your medical record has been accessed by:</AttributeValue>
1487 [h125]             </AttributeAssignmentExpression>
1488 [h126]         <AttributeAssignmentExpression
1489 [h127]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
1490 [h128]             <AttributeDesignator
1491 [h129]                 MustBePresent="false"
1492 [h130]                 Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1493 [h131]                 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
1494 [h132]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1495 [h133]             </AttributeAssignmentExpression>
1496 [h134]         </ObligationExpression>
1497 [h135]     </ObligationExpressions>
1498 [h136] </Policy>

```

1499 [h2] - [h10] The <Policy> element includes standard namespace declarations as well as **policy** specific  
1500 parameters, such as PolicyId and RuleCombiningAlgId.

1501 [h8] **Policy** identifier. This parameter allows the **policy** to be referenced by a **policy set**.

1502 [h10] The **Rule-combining algorithm** identifies the algorithm for combining the outcomes of **rule**  
1503 evaluation.

1504 [h11] - [h14] Free-form description of the **policy**.

1505 [h18] - [h33] **Policy target**. The **policy target** defines a set of applicable **decision requests**. The  
1506 structure of the <Target> element in the <Policy> is identical to the structure of the <Target>

1507 element in the <Rule>. In this case, the **policy target** is the set of all XML **resources** that conform to  
 1508 the namespace “urn:example:med:schemas:record”.

1509 [h34] - [h108] The only <Rule> element included in this <Policy>. Two parameters are specified in the  
 1510 **rule** header: RuleId and Effect.

1511 [h41] - [h87] The **rule target** further constrains the **policy target**.

1512 [h44] - [h53] The <Match> element targets the **rule** at **subjects** whose  
 1513 “urn:oasis:names:tc:xacml:3.0:example:attribute:role” **subject attribute** is equal to “physician”.

1514 [h58] - [h69] The <Match> element targets the **rule** at **resources** that match the XPath expression  
 1515 “md:record/md:medical”.

1516 [h74] - [h84] The <Match> element targets the **rule** at **actions** whose  
 1517 “urn:oasis:names:tc:xacml:1.0:action:action-id” **action attribute** is equal to “write”.

1518 [h88] - [h107] The <Condition> element. For the **rule** to be applicable to the **decision request**, the  
 1519 **condition** must evaluate to “True”. This **condition** compares the value of the  
 1520 “urn:oasis:names:tc:xacml:3.0:example:attribute:physician-id” **subject attribute** with the value of the  
 1521 <registrationId> element in the medical record that is being accessed.

1522 [h109] - [h134] The <ObligationExpressions> element. **Obligations** are a set of operations that  
 1523 must be performed by the **PEP** in conjunction with an **authorization decision**. An **obligation** may be  
 1524 associated with a “Permit” or “Deny” **authorization decision**. The element contains a single **obligation**  
 1525 expression, which will be evaluated into an obligation when the policy is evaluated.

1526 [h110] - [h133] The <ObligationExpression> element consists of the ObligationId attribute, the  
 1527 **authorization decision** value for which it must be fulfilled, and a set of **attribute** assignments.

1528 [h110] The ObligationId attribute identifies the **obligation**. In this case, the **PEP** is required to send  
 1529 email.

1530 [h111] The FulfillOn attribute defines the **authorization decision** value for which the **obligation**  
 1531 derived from the **obligation** expression must be fulfilled. In this case, the **obligation** must be fulfilled  
 1532 when **access** is permitted.

1533 [h112] - [h119] The first parameter indicates where the **PEP** will find the email address in the **resource**.  
 1534 The **PDP** will evaluate the <AttributeSelector> and return the result to the **PEP** inside the resulting  
 1535 **obligation**.

1536 [h120] - [h123] The second parameter contains literal text for the email body.

1537 [h125] - [h132] The third parameter indicates where the **PEP** will find further text for the email body in the  
 1538 **resource**. The **PDP** will evaluate the <AttributeDesignator> and return the result to the **PEP** inside  
 1539 the resulting **obligation**.

#### 1540 4.2.4.4 Rule 4

1541 **Rule 4** illustrates the use of the “Deny” **Effect** value, and a <Rule> with no <Condition> element.

```

1542 [i1] <?xml version="1.0" encoding="UTF-8"?>
1543 [i2] <Policy
1544 [i3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1545 [i4]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1546 [i5]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1547 [i6]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:4"
1548 [i7]   Version="1.0"
1549 [i8]   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1550 [i9]   algorithm:deny-overrides">
1551 [i9]   <PolicyDefaults>
1552 [i10]     <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1553 [i11]   </PolicyDefaults>
1554 [i12]   <Target/>
1555 [i13]   <Rule
1556 [i14]     RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:4"
1557 [i15]     Effect="Deny">
1558 [i16]   <Description>
1559 [i17]     An Administrator shall not be permitted to read or write

```

```

1560 [i18] medical elements of a patient record in the
1561 [i19] http://www.med.example.com/records.xsd namespace.
1562 [i20] </Description>
1563 [i21] <Target>
1564 [i22] <AnyOf>
1565 [i23] <AllOf>
1566 [i24] <Match
1567 [i25] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1568 [i26] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1569 [i27] >administrator</AttributeValue>
1570 [i28] <AttributeDesignator
1571 [i29] MustBePresent="false"
1572 [i30] Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1573 [i31] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
1574 [i32] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1575 [i33] </Match>
1576 [i34] </AllOf>
1577 [i35] </AnyOf>
1578 [i36] <AnyOf>
1579 [i37] <AllOf>
1580 [i38] <Match
1581 [i39] MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1582 [i40] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1583 [i41] >urn:example:med:schemas:record</AttributeValue>
1584 [i42] <AttributeDesignator
1585 [i43] MustBePresent="false"
1586 [i44] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1587 [i45] AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1588 [i46] DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1589 [i47] </Match>
1590 [i48] <Match
1591 [i49] MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1592 [i50] <AttributeValue
1593 [i51] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1594 [i52] XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1595 [i53] >md:record/md:medical</AttributeValue>
1596 [i54] <AttributeDesignator
1597 [i55] MustBePresent="false"
1598 [i56] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1599 [i57] AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1600 [i58] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1601 [i59] </Match>
1602 [i60] </AllOf>
1603 [i61] </AnyOf>
1604 [i62] <AnyOf>
1605 [i63] <AllOf>
1606 [i64] <Match
1607 [i65] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1608 [i66] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1609 [i67] >read</AttributeValue>
1610 [i68] <AttributeDesignator
1611 [i69] MustBePresent="false"
1612 [i70] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1613 [i71] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1614 [i72] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1615 [i73] </Match>
1616 [i74] </AllOf>
1617 [i75] <AllOf>
1618 [i76] <Match
1619 [i77] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1620 [i78] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1621 [i79] >write</AttributeValue>
1622 [i80] <AttributeDesignator
1623 [i81] MustBePresent="false"
1624 [i82] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1625 [i83] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1626 [i84] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1627 [i85] </Match>
1628 [i86] </AllOf>
1629 [i87] </AnyOf>
1630 [i88] </Target>
1631 [i89] </Rule>
1632 [i90] </Policy>

```



1633 [i13] - [i15] The <Rule> element declaration.

1634 [i15] **Rule Effect**. Every **rule** that evaluates to “True” emits the **rule effect** as its value. This **rule**  
 1635 **Effect** is “Deny” meaning that according to this **rule**, **access** must be denied when it evaluates to  
 1636 “True”.

1637 [i16] - [i20] Free form description of the **rule**.

1638 [i21] - [i88] **Rule target**. The **Rule target** defines the set of **decision requests** that are applicable to the  
 1639 **rule**.

1640 [i24] - [i33] The <Match> element targets the **rule** at **subjects** whose  
 1641 “urn:oasis:names:tc:xacml:3.0:example:attribute:role” **subject attribute** is equal to “administrator”.

1642 [i36] - [i61] The <AnyOf> element contains one <AllOf> element, which (in turn) contains two <Match>  
 1643 elements. The **target** matches if the **resource** identified by the request **context** matches both **resource**  
 1644 match criteria.

1645 [i38] - [i47] The first <Match> element targets the **rule** at **resources** whose  
 1646 “urn:oasis:names:tc:xacml:2.0:resource:target-namespace” **resource attribute** is equal to  
 1647 “urn:example:med:schemas:record”.

1648 [i48] - [i59] The second <Match> element targets the **rule** at XML elements that match the XPath  
 1649 expression “/md:record/md:medical”.

1650 [i62] - [i87] The <AnyOf> element contains two <AllOf> elements, each of which contains one <Match>  
 1651 element. The **target** matches if the **action** identified in the request **context** matches either of the **action**  
 1652 match criteria.

1653 [i64] - [i85] The <Match> elements **target** the **rule** at **actions** whose  
 1654 “urn:oasis:names:tc:xacml:1.0:action:action-id” **action attribute** is equal to “read” or “write”.

1655 This **rule** does not have a <Condition> element.

#### 1656 4.2.4.5 Example PolicySet

1657 This section uses the examples of the previous sections to illustrate the process of combining **policies**.  
 1658 The **policy** governing read **access** to medical elements of a record is formed from each of the four **rules**  
 1659 described in Section 4.2.3. In plain language, the combined **rule** is:

- 1660 • Either the requestor is the patient; or
- 1661 • the requestor is the parent or guardian and the patient is under 16; or
- 1662 • the requestor is the primary care physician and a notification is sent to the patient; and
- 1663 • the requestor is not an administrator.

1664 The following **policy set** illustrates the combined **policies**. **Policy 3** is included by reference and **policy**  
 1665 **2** is explicitly included.

```

1666 [j1]    <?xml version="1.0" encoding="UTF-8"?>
1667 [j2]    <PolicySet
1668 [j3]      xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1669 [j4]      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1670 [j5]      PolicySetId="urn:oasis:names:tc:xacml:3.0:example:policysetid:1"
1671 [j6]      Version="1.0"
1672 [j7]      PolicyCombiningAlgId=
1673 [j8]      "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides">
1674 [j9]      <Description>
1675 [j10]        Example policy set.
1676 [j11]      </Description>
1677 [j12]      <Target>
1678 [j13]        <AnyOf>
1679 [j14]          <AllOf>
1680 [j15]            <Match
1681 [j16]              MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1682 [j17]                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1683 [j18]                  >urn:example:med:schema:records</AttributeValue>
1684 [j19]                <AttributeDesignator
1685 [j20]                  MustBePresent="false"

```

```

1686 [j21]         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1687 [j22]         AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1688 [j23]         DataType="http://www.w3.org/2001/XMLSchema#string"/>
1689 [j24]         </Match>
1690 [j25]         </AllOf>
1691 [j26]         </AnyOf>
1692 [j27]         </Target>
1693 [j28]         <PolicyIdReference>
1694 [j29]           urn:oasis:names:tc:xacml:3.0:example:policyid:3
1695 [j30]         </PolicyIdReference>
1696 [j31]         <Policy
1697 [j32]           PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"
1698 [j33]           RuleCombiningAlgId=
1699 [j34]             "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides"
1700 [j35]           Version="1.0">
1701 [j36]           <Target/>
1702 [j37]           <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"
1703 [j38]             Effect="Permit">
1704 [j39]           </Rule>
1705 [j40]           <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"
1706 [j41]             Effect="Permit">
1707 [j42]           </Rule>
1708 [j43]           <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:4"
1709 [j44]             Effect="Deny">
1710 [j45]           </Rule>
1711 [j46]         </Policy>
1712 [j47]         </PolicySet>

```

1713 [j2] - [j8] The <PolicySet> element declaration. Standard XML namespace declarations are included.

1714 [j5] The PolicySetId attribute is used for identifying this **policy set** for possible inclusion in another  
1715 **policy set**.

1716 [j7] - [j8] The **policy-combining algorithm** identifier. **Policies** and **policy sets** in this **policy set** are  
1717 combined according to the specified **policy-combining algorithm** when the **authorization decision** is  
1718 computed.

1719 [j9] - [j11] Free form description of the **policy set**.

1720 [j12] - [j27] The **policy set** <Target> element defines the set of **decision requests** that are applicable to  
1721 this <PolicySet> element.

1722 [j28] - [j30] PolicyIdReference includes a **policy** by id.

1723 [j31] - [j46] **Policy 2** is explicitly included in this **policy set**. The **rules** in **Policy 2** are omitted for clarity.

---

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

### 5.1 Element <PolicySet>

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

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

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

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

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

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

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

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

```
<xs:element name="PolicySet" type="xacml:PolicySetType"/>
<xs:complexType name="PolicySetType">
  <xs:sequence>
    <xs:element ref="xacml:Description" minOccurs="0"/>
    <xs:element ref="xacml:PolicyIssuer" minOccurs="0"/>
    <xs:element ref="xacml:PolicySetDefaults" minOccurs="0"/>
    <xs:element ref="xacml:Target"/>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="xacml:PolicySet"/>
      <xs:element ref="xacml:Policy"/>
      <xs:element ref="xacml:PolicySetIdReference"/>
      <xs:element ref="xacml:PolicyIdReference"/>
      <xs:element ref="xacml:CombinerParameters"/>
      <xs:element ref="xacml:PolicyCombinerParameters"/>
      <xs:element ref="xacml:PolicySetCombinerParameters"/>
    </xs:choice>
    <xs:element ref="xacml:ObligationExpressions" minOccurs="0"/>
    <xs:element ref="xacml:AdviceExpressions" minOccurs="0"/>
  </xs:sequence>
```

```

1773 <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
1774 <xs:attribute name="Version" type="xacml:VersionType" use="required"/>
1775 <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
1776 <xs:attribute name="MaxDelegationDepth" type="xs:integer" use="optional"/>
1777 </xs:complexType>

```

1778 The <PolicySet> element is of PolicySetType complex type.

1779 The <PolicySet> element contains the following attributes and elements:

1780 PolicySetId [Required]

1781 **Policy set** identifier. It is the responsibility of the **PAP** to ensure that no two **policies** visible to  
1782 the **PDP** have the same identifier. This MAY be achieved by following a predefined URN or URI  
1783 scheme. If the **policy set** identifier is in the form of a URL, then it MAY be resolvable.

1784 Version [Required]

1785 The version number of the PolicySet.

1786 PolicyCombiningAlgId [Required]

1787 The identifier of the **policy-combining algorithm** by which the <PolicySet>,  
1788 <CombinerParameters>, <PolicyCombinerParameters> and  
1789 <PolicySetCombinerParameters> components MUST be combined. Standard **policy-**  
1790 **combining algorithms** are listed in Appendix Appendix C. Standard **policy-combining**  
1791 **algorithm** identifiers are listed in Section B.9.

1792 MaxDelegationDepth [Optional]

1793 If present, limits the depth of delegation which is authorized by this **policy set**. See the delegation  
1794 profile [XACMLAdmin].

1795 <Description> [Optional]

1796 A free-form description of the **policy set**.

1797 <PolicyIssuer> [Optional]

1798 **Attributes** of the **issuer** of the **policy set**.

1799 <PolicySetDefaults> [Optional]

1800 A set of default values applicable to the **policy set**. The scope of the <PolicySetDefaults>  
1801 element SHALL be the enclosing **policy set**.

1802 <Target> [Required]

1803 The <Target> element defines the applicability of a **policy set** to a set of **decision requests**.

1804 The <Target> element MAY be declared by the creator of the <PolicySet> or it MAY be computed  
1805 from the <Target> elements of the referenced <Policy> elements, either as an intersection or  
1806 as a union.

1807 <PolicySet> [Any Number]

1808 A **policy set** that is included in this **policy set**.

1809 <Policy> [Any Number]

1810 A **policy** that is included in this **policy set**.

1811 <PolicySetIdReference> [Any Number]

1812 A reference to a **policy set** that MUST be included in this **policy set**. If  
1813 <PolicySetIdReference> is a URL, then it MAY be resolvable.

1814 <PolicyIdReference> [Any Number]

1815 A reference to a **policy** that MUST be included in this **policy set**. If the  
1816 <PolicyIdReference> is a URL, then it MAY be resolvable.

- 1817 <ObligationExpressions> [Optional]  
 1818       Contains the set of <ObligationExpression> elements. See Section 7.18 for a description of  
 1819       how the set of **obligations** to be returned by the **PDP** shall be determined.
- 1820 <AdviceExpressions> [Optional]  
 1821       Contains the set of <AdviceExpression> elements. See Section 7.18 for a description of how  
 1822       the set of **advice** to be returned by the **PDP** shall be determined.
- 1823 <CombinerParameters> [Optional]  
 1824       Contains a sequence of <CombinerParameter> elements. The parameters apply to the  
 1825       combining algorithm as such and it is up to the specific combining algorithm to interpret them and  
 1826       adjust its behavior accordingly.
- 1827 <PolicyCombinerParameters> [Optional]  
 1828       Contains a sequence of <CombinerParameter> elements that are associated with a particular  
 1829       <Policy> or <PolicyIdReference> element within the <PolicySet>. It is up to the specific  
 1830       combining algorithm to interpret them and adjust its behavior accordingly.
- 1831 <PolicySetCombinerParameters> [Optional]  
 1832       Contains a sequence of <CombinerParameter> elements that are associated with a particular  
 1833       <PolicySet> or <PolicySetIdReference> element within the <PolicySet>. It is up to the  
 1834       specific combining algorithm to interpret them and adjust its behavior accordingly.

## 1835 5.2 Element <Description>

- 1836 The <Description> element contains a free-form description of the <PolicySet>, <Policy>,  
 1837 <Rule> or <Apply> element. The <Description> element is of xs:string simple type.

```
1838 <xs:element name="Description" type="xs:string"/>
```

## 1839 5.3 Element <PolicyIssuer>

- 1840 The <PolicyIssuer> element contains **attributes** describing the issuer of the **policy** or **policy set**.  
 1841 The use of the **policy** issuer element is defined in a separate administration profile [**XACMLAdmin**]. A  
 1842 PDP which does not implement the administration profile MUST report an error or return an Indeterminate  
 1843 result if it encounters this element.

```
1844 <xs:element name="PolicyIssuer" type="xacml:PolicyIssuerType"/>
1845 <xs:complexType name="PolicyIssuerType">
1846   <xs:sequence>
1847     <xs:element ref="xacml:Content" minOccurs="0"/>
1848     <xs:element ref="xacml:Attribute" minOccurs="0" maxOccurs="unbounded"/>
1849   </xs:sequence>
1850 </xs:complexType>
```

- 1851 The <PolicyIssuer> element is of PolicyIssuerType complex type.

- 1852 The <PolicyIssuer> element contains the following elements:

1853 <Content> [Optional]

- 1854       Free form XML describing the issuer. See Section 5.45.

1855 <Attribute> [Zero to many]

- 1856       An **attribute** of the issuer. See Section 5.46.

## 1857 5.4 Element <PolicySetDefaults>

- 1858 The <PolicySetDefaults> element SHALL specify default values that apply to the <PolicySet>  
 1859 element.

```

1860 <xs:element name="PolicySetDefaults" type="xacml:DefaultsType"/>
1861 <xs:complexType name="DefaultsType">
1862   <xs:sequence>
1863     <xs:choice>
1864       <xs:element ref="xacml:XPathVersion">
1865     </xs:choice>
1866   </xs:sequence>
1867 </xs:complexType>

```

1868 <PolicySetDefaults> element is of DefaultsType complex type.

1869 The <PolicySetDefaults> element contains the following elements:

1870 <XPathVersion> [Optional]

1871       Default XPath version.

## 1872 5.5 Element <XPathVersion>

1873 The <XPathVersion> element SHALL specify the version of the XPath specification to be used by  
1874 <AttributeSelector> elements and XPath-based functions in the **policy set** or **policy**.

```

1875 <xs:element name="XPathVersion" type="xs:anyURI"/>

```

1876 The URI for the XPath 1.0 specification is "http://www.w3.org/TR/1999/REC-xpath-19991116".

1877 The URI for the XPath 2.0 specification is "http://www.w3.org/TR/2007/REC-xpath20-20070123".

1878 The <XPathVersion> element is REQUIRED if the XACML enclosing **policy set** or **policy** contains  
1879 <AttributeSelector> elements or XPath-based functions.

## 1880 5.6 Element <Target>

1881 The <Target> element identifies the set of **decision requests** that the parent element is intended to  
1882 evaluate. The <Target> element SHALL appear as a child of a <PolicySet> and <Policy> element  
1883 and MAY appear as a child of a <Rule> element.

1884 The <Target> element SHALL contain a **conjunctive sequence** of <AnyOf> elements. For the parent  
1885 of the <Target> element to be applicable to the **decision request**, there MUST be at least one positive  
1886 match between each <AnyOf> element of the <Target> element and the corresponding section of the  
1887 <Request> element.

```

1888 <xs:element name="Target" type="xacml:TargetType"/>
1889 <xs:complexType name="TargetType">
1890   <xs:sequence minOccurs="0" maxOccurs="unbounded">
1891     <xs:element ref="xacml:AnyOf"/>
1892   </xs:sequence>
1893 </xs:complexType>

```

1894 The <Target> element is of TargetType complex type.

1895 The <Target> element contains the following elements:

1896 <AnyOf> [Zero to Many]

1897       Matching specification for **attributes** in the **context**. If this element is missing, then the **target**  
1898       SHALL match all **contexts**.

## 1899 5.7 Element <AnyOf>

1900 The <AnyOf> element SHALL contain a **disjunctive sequence** of <AllOf> elements.

```

1901 <xs:element name="AnyOf" type="xacml:AnyOfType"/>
1902 <xs:complexType name="AnyOfType">
1903   <xs:sequence minOccurs="1" maxOccurs="unbounded">
1904     <xs:element ref="xacml:AllOf"/>

```

1905 `</xs:sequence>`  
1906 `</xs:complexType>`

1907 The `<AnyOf>` element is of `AnyOfType` complex type.

1908 The `<AnyOf>` element contains the following elements:

1909 `<AllOf>` [One to Many, Required]

1910 See Section 5.8.

## 1911 5.8 Element `<AllOf>`

1912 The `<AllOf>` element SHALL contain a **conjunctive sequence** of `<Match>` elements.

```
1913 <xs:element name="AllOf" type="xacml:AllOfType"/>
1914 <xs:complexType name="AllOfType">
1915   <xs:sequence minOccurs="1" maxOccurs="unbounded">
1916     <xs:element ref="xacml:Match"/>
1917   </xs:sequence>
1918 </xs:complexType>
```

1919 The `<AllOf>` element is of `AllOfType` complex type.

1920 The `<AllOf>` element contains the following elements:

1921 `<Match>` [One to Many]

1922 A **conjunctive sequence** of individual matches of the **attributes** in the request **context** and the  
1923 embedded **attribute** values. See Section 5.9.

## 1924 5.9 Element `<Match>`

1925 The `<Match>` element SHALL identify a set of entities by matching **attribute** values in an

1926 `<Attributes>` element of the request **context** with the embedded **attribute** value.

```
1927 <xs:element name="Match" type="xacml:MatchType"/>
1928 <xs:complexType name="MatchType">
1929   <xs:sequence>
1930     <xs:element ref="xacml:AttributeValue"/>
1931     <xs:choice>
1932       <xs:element ref="xacml:AttributeDesignator"/>
1933       <xs:element ref="xacml:AttributeSelector"/>
1934     </xs:choice>
1935   </xs:sequence>
1936   <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
1937 </xs:complexType>
```

1938 The `<Match>` element is of `MatchType` complex type.

1939 The `<Match>` element contains the following attributes and elements:

1940 `MatchId` [Required]

1941 Specifies a matching function. The value of this attribute MUST be of type `xs:anyURI` with legal  
1942 values documented in Section 7.6.

1943 `<AttributeValue>` [Required]

1944 Embedded **attribute** value.

1945 `<AttributeDesignator>` [Required choice]

1946 MAY be used to identify one or more **attribute** values in an `<Attributes>` element of the  
1947 request **context**.

1948 `<AttributeSelector>` [Required choice]



1949 MAY be used to identify one or more **attribute** values in a <Content> element of the request  
1950 **context**.

## 1951 5.10 Element <PolicySetIdReference>

1952 The <PolicySetIdReference> element SHALL be used to reference a <PolicySet> element by id.  
1953 If <PolicySetIdReference> is a URL, then it MAY be resolvable to the <PolicySet> element.  
1954 However, the mechanism for resolving a **policy set** reference to the corresponding **policy set** is outside  
1955 the scope of this specification.

```
1956 <xs:element name="PolicySetIdReference" type="xacml:IdReferenceType"/>  
1957 <xs:complexType name="IdReferenceType">  
1958   <xs:simpleContent>  
1959     <xs:extension base="xs:anyURI">  
1960       <xs:attribute name="xacml:Version"  
1961         type="xacml:VersionMatchType" use="optional"/>  
1962       <xs:attribute name="xacml:EarliestVersion"  
1963         type="xacml:VersionMatchType" use="optional"/>  
1964       <xs:attribute name="xacml:LatestVersion"  
1965         type="xacml:VersionMatchType" use="optional"/>  
1966     </xs:extension>  
1967   </xs:simpleContent>  
1968 </xs:complexType>
```

1969 Element <PolicySetIdReference> is of xacml:IdReferenceType complex type.

1970 IdReferenceType extends the xs:anyURI type with the following attributes:

1971 Version [Optional]

1972 Specifies a matching expression for the version of the **policy set** referenced.

1973 EarliestVersion [Optional]

1974 Specifies a matching expression for the earliest acceptable version of the **policy set** referenced.

1975 LatestVersion [Optional]

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

1977 The matching operation is defined in Section 5.13. Any combination of these attributes MAY be present  
1978 in a <PolicySetIdReference>. The referenced **policy set** MUST match all expressions. If none of  
1979 these attributes is present, then any version of the **policy set** is acceptable. In the case that more than  
1980 one matching version can be obtained, then the most recent one SHOULD be used.

## 1981 5.11 Element <PolicyIdReference>

1982 The <PolicyIdReference> element SHALL be used to reference a <Policy> element by id. If  
1983 <PolicyIdReference> is a URL, then it MAY be resolvable to the <Policy> element. However, the  
1984 mechanism for resolving a **policy** reference to the corresponding **policy** is outside the scope of this  
1985 specification.

```
1986 <xs:element name="PolicyIdReference" type="xacml:IdReferenceType"/>
```

1987 Element <PolicyIdReference> is of xacml:IdReferenceType complex type (see Section 5.10) .

## 1988 5.12 Simple type VersionType

1989 Elements of this type SHALL contain the version number of the **policy** or **policy set**.

```
1990 <xs:simpleType name="VersionType">  
1991   <xs:restriction base="xs:string">  
1992     <xs:pattern value="(\d+\.)*\d+"/>  
1993   </xs:restriction>  
1994 </xs:simpleType>
```



1995 The version number is expressed as a sequence of decimal numbers, each separated by a period (.).  
1996 'd+' represents a sequence of one or more decimal digits.

### 1997 5.13 Simple type VersionMatchType

1998 Elements of this type SHALL contain a restricted regular expression matching a version number (see  
1999 Section 5.12). The expression SHALL match versions of a referenced **policy** or **policy set** that are  
2000 acceptable for inclusion in the referencing **policy** or **policy set**.

```
2001 <xs:simpleType name="VersionMatchType">  
2002   <xs:restriction base="xs:string">  
2003     <xs:pattern value="((\d+|\*)\.)*(\d+|\*|\+)" />  
2004   </xs:restriction>  
2005 </xs:simpleType>
```

2006 A version match is '.'-separated, like a version string. A number represents a direct numeric match. A '\*'  
2007 means that any single number is valid. A '+' means that any number, and any subsequent numbers, are  
2008 valid. In this manner, the following four patterns would all match the version string '1.2.3': '1.2.3', '1.\*.3',  
2009 '1.2.\*' and '1.+'

### 2010 5.14 Element <Policy>

2011 The <Policy> element is the smallest entity that SHALL be presented to the **PDP** for evaluation.

2012 A <Policy> element may be evaluated, in which case the evaluation procedure defined in Section 7.12  
2013 SHALL be used.

2014 The main components of this element are the <Target>, <Rule>, <CombinerParameters>,  
2015 <RuleCombinerParameters>, <ObligationExpressions> and <AdviceExpressions>  
2016 elements and the RuleCombiningAlgId attribute.

2017 A <Policy> element MAY contain a <PolicyIssuer> element. The interpretation of the  
2018 <PolicyIssuer> element is explained in the separate administrative **policy** profile [XACMLAdmin].

2019 The <Target> element defines the applicability of the <Policy> element to a set of **decision requests**.  
2020 If the <Target> element within the <Policy> element matches the request **context**, then the  
2021 <Policy> element MAY be used by the **PDP** in making its **authorization decision**. See Section 7.12.

2022 The <Policy> element includes a sequence of choices between <VariableDefinition> and  
2023 <Rule> elements.

2024 **Rules** included in the <Policy> element MUST be combined by the algorithm specified by the  
2025 RuleCombiningAlgId attribute.

2026 The <ObligationExpressions> element contains a set of **obligation** expressions that MUST be  
2027 evaluated into **obligations** by the **PDP** and the resulting **obligations** MUST be fulfilled by the **PEP** in  
2028 conjunction with the **authorization decision**. If the **PEP** does not understand, or cannot fulfill, any of the  
2029 **obligations**, then it MUST act according to the PEP bias. See Section 7.2 and 7.18.

2030 The <AdviceExpressions> element contains a set of **advice** expressions that MUST be evaluated into  
2031 **advice** by the **PDP**. The resulting **advice** MAY be safely ignored by the **PEP** in conjunction with the  
2032 **authorization decision**. See Section 7.18.

```
2033 <xs:element name="Policy" type="xacml:PolicyType"/>  
2034 <xs:complexType name="PolicyType">  
2035   <xs:sequence>  
2036     <xs:element ref="xacml:Description" minOccurs="0"/>  
2037     <xs:element ref="xacml:PolicyIssuer" minOccurs="0"/>  
2038     <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>  
2039     <xs:element ref="xacml:Target"/>  
2040     <xs:choice maxOccurs="unbounded">  
2041       <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>  
2042       <xs:element ref="xacml:RuleCombinerParameters" minOccurs="0"/>  
2043       <xs:element ref="xacml:VariableDefinition"/>
```

```

2044         <xs:element ref="xacml:Rule"/>
2045     </xs:choice>
2046     <xs:element ref="xacml:ObligationExpressions" minOccurs="0"/>
2047     <xs:element ref="xacml:AdviceExpressions" minOccurs="0"/>
2048 </xs:sequence>
2049 <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
2050 <xs:attribute name="Version" type="xacml:VersionType" use="required"/>
2051 <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
2052 <xs:attribute name="MaxDelegationDepth" type="xs:integer" use="optional"/>
2053 </xs:complexType>

```

2054 The <Policy> element is of PolicyType complex type.

2055 The <Policy> element contains the following attributes and elements:

2056 PolicyId [Required]

2057 **Policy** identifier. It is the responsibility of the **PAP** to ensure that no two **policies** visible to the  
2058 **PD**P have the same identifier. This MAY be achieved by following a predefined URN or URI  
2059 scheme. If the **policy** identifier is in the form of a URL, then it MAY be resolvable.

2060 Version [Required]

2061 The version number of the **Policy**.

2062 RuleCombiningAlgId [Required]

2063 The identifier of the **rule-combining algorithm** by which the <Policy>,  
2064 <CombinerParameters> and <RuleCombinerParameters> components MUST be  
2065 combined. Standard **rule-combining algorithms** are listed in Appendix Appendix C. Standard  
2066 **rule-combining algorithm** identifiers are listed in Section B.9.

2067 MaxDelegationDepth [Optional]

2068 If present, limits the depth of delegation which is authorized by this **policy**. See the delegation  
2069 profile [XACMLAdmin].

2070 <Description> [Optional]

2071 A free-form description of the **policy**. See Section 5.2.

2072 <PolicyIssuer> [Optional]

2073 **Attributes** of the **issuer** of the **policy**.

2074 <PolicyDefaults> [Optional]

2075 Defines a set of default values applicable to the **policy**. The scope of the <PolicyDefaults>  
2076 element SHALL be the enclosing **policy**.

2077 <CombinerParameters> [Optional]

2078 A sequence of parameters to be used by the **rule-combining algorithm**. The parameters apply  
2079 to the combining algorithm as such and it is up to the specific combining algorithm to interpret  
2080 them and adjust its behavior accordingly.

2081 <RuleCombinerParameters> [Optional]

2082 A sequence of <RuleCombinerParameter> elements that are associated with a particular  
2083 <Rule> element within the <Policy>.. It is up to the specific combining algorithm to interpret  
2084 them and adjust its behavior accordingly.

2085 <Target> [Required]

2086 The <Target> element defines the applicability of a <Policy> to a set of **decision requests**.

2087 The <Target> element MAY be declared by the creator of the <Policy> element, or it MAY be  
2088 computed from the <Target> elements of the referenced <Rule> elements either as an  
2089 intersection or as a union.

2090 <VariableDefinition> [Any Number]  
 2091 Common function definitions that can be referenced from anywhere in a *rule* where an  
 2092 expression can be found.

2093 <Rule> [Any Number]  
 2094 A sequence of *rules* that MUST be combined according to the RuleCombiningAlgId attribute.  
 2095 *Rules* whose <Target> elements and conditions match the *decision request* MUST be  
 2096 considered. *Rules* whose <Target> elements or conditions do not match the *decision request*  
 2097 SHALL be ignored.

2098 <ObligationExpressions> [Optional]  
 2099 A *conjunctive sequence* of *obligation* expressions which MUST be evaluated into *obligations*  
 2100 by the PDP. The corresponding *obligations* MUST be fulfilled by the *PEP* in conjunction with the  
 2101 *authorization decision*. See Section 7.18 for a description of how the set of *obligations* to be  
 2102 returned by the *PDP* SHALL be determined. See section 7.2 about enforcement of *obligations*.

2103 <AdviceExpressions> [Optional]  
 2104 A *conjunctive sequence* of *advice* expressions which MUST evaluated into *advice* by the *PDP*.  
 2105 The corresponding *advice* provide supplementary information to the *PEP* in conjunction with the  
 2106 *authorization decision*. See Section 7.18 for a description of how the set of *advice* to be  
 2107 returned by the *PDP* SHALL be determined.

## 2108 5.15 Element <PolicyDefaults>

2109 The <PolicyDefaults> element SHALL specify default values that apply to the <Policy> element.

```
2110 <xs:element name="PolicyDefaults" type="xacml:DefaultsType"/>
2111 <xs:complexType name="DefaultsType">
2112   <xs:sequence>
2113     <xs:choice>
2114       <xs:element ref="xacml:XPathVersion" />
2115     </xs:choice>
2116   </xs:sequence>
2117 </xs:complexType>
```

2118 <PolicyDefaults> element is of DefaultsType complex type.

2119 The <PolicyDefaults> element contains the following elements:

2120 <XPathVersion> [Optional]

2121 Default XPath version.

## 2122 5.16 Element <CombinerParameters>

2123 The <CombinerParameters> element conveys parameters for a *policy-* or *rule-combining algorithm*.

2124 If multiple <CombinerParameters> elements occur within the same *policy* or *policy set*, they SHALL  
 2125 be considered equal to one <CombinerParameters> element containing the concatenation of all the  
 2126 sequences of <CombinerParameters> contained in all the aforementioned <CombinerParameters>  
 2127 elements, such that the order of occurrence of the <CombinerParameters> elements is preserved in  
 2128 the concatenation of the <CombinerParameter> elements.

2129 Note that none of the combining algorithms specified in XACML 3.0 is parameterized.

```
2130 <xs:element name="CombinerParameters" type="xacml:CombinerParametersType"/>
2131 <xs:complexType name="CombinerParametersType">
2132   <xs:sequence>
2133     <xs:element ref="xacml:CombinerParameter" minOccurs="0"
2134       maxOccurs="unbounded"/>
2135   </xs:sequence>
```

2136 `</xs:complexType>`

2137 The `<CombinerParameters>` element is of `CombinerParametersType` complex type.

2138 The `<CombinerParameters>` element contains the following elements:

2139 `<CombinerParameter>` [Any Number]

2140 A single parameter. See Section 5.17.

2141 Support for the `<CombinerParameters>` element is optional.

## 2142 5.17 Element `<CombinerParameter>`

2143 The `<CombinerParameter>` element conveys a single parameter for a *policy- or rule-combining*  
2144 *algorithm*.

```
2145 <xs:element name="CombinerParameter" type="xacml:CombinerParameterType"/>
2146 <xs:complexType name="CombinerParameterType">
2147   <xs:sequence>
2148     <xs:element ref="xacml:AttributeValue"/>
2149   </xs:sequence>
2150   <xs:attribute name="ParameterName" type="xs:string" use="required"/>
2151 </xs:complexType>
```

2152 The `<CombinerParameter>` element is of `CombinerParameterType` complex type.

2153 The `<CombinerParameter>` element contains the following attributes:

2154 `ParameterName` [Required]

2155 The identifier of the parameter.

2156 `<AttributeValue>` [Required]

2157 The value of the parameter.

2158 Support for the `<CombinerParameter>` element is optional.

## 2159 5.18 Element `<RuleCombinerParameters>`

2160 The `<RuleCombinerParameters>` element conveys parameters associated with a particular *rule*  
2161 within a *policy* for a *rule-combining algorithm*.

2162 Each `<RuleCombinerParameters>` element MUST be associated with a *rule* contained within the  
2163 same *policy*. If multiple `<RuleCombinerParameters>` elements reference the same *rule*, they SHALL  
2164 be considered equal to one `<RuleCombinerParameters>` element containing the concatenation of all  
2165 the sequences of `<CombinerParameters>` contained in all the aforementioned  
2166 `<RuleCombinerParameters>` elements, such that the order of occurrence of the  
2167 `<RuleCombinerParameters>` elements is preserved in the concatenation of the  
2168 `<CombinerParameter>` elements.

2169 Note that none of the *rule-combining algorithms* specified in XACML 3.0 is parameterized.

```
2170 <xs:element name="RuleCombinerParameters"
2171   type="xacml:RuleCombinerParametersType"/>
2172 <xs:complexType name="RuleCombinerParametersType">
2173   <xs:complexContent>
2174     <xs:extension base="xacml:CombinerParametersType">
2175       <xs:attribute name="RuleIdRef" type="xs:string"
2176         use="required"/>
2177     </xs:extension>
2178   </xs:complexContent>
2179 </xs:complexType>
```

2180 The `<RuleCombinerParameters>` element contains the following attribute:

2181 RuleIdRef [Required]  
2182       The identifier of the <Rule> contained in the *policy*.  
2183 Support for the <RuleCombinerParameters> element is optional, only if support for combiner  
2184 parameters is not implemented.

## 2185 **5.19 Element <PolicyCombinerParameters>**

2186 The <PolicyCombinerParameters> element conveys parameters associated with a particular *policy*  
2187 within a *policy set* for a *policy-combining algorithm*.

2188 Each <PolicyCombinerParameters> element MUST be associated with a *policy* contained within the  
2189 same *policy set*. If multiple <PolicyCombinerParameters> elements reference the same *policy*,  
2190 they SHALL be considered equal to one <PolicyCombinerParameters> element containing the  
2191 concatenation of all the sequences of <CombinerParameters> contained in all the aforementioned  
2192 <PolicyCombinerParameters> elements, such that the order of occurrence of the  
2193 <PolicyCombinerParameters> elements is preserved in the concatenation of the  
2194 <CombinerParameter> elements.

2195 Note that none of the *policy-combining algorithms* specified in XACML 3.0 is parameterized.

```
2196 <xs:element name="PolicyCombinerParameters"  
2197 type="xacml:PolicyCombinerParametersType"/>  
2198 <xs:complexType name="PolicyCombinerParametersType">  
2199   <xs:complexContent>  
2200     <xs:extension base="xacml:CombinerParametersType">  
2201       <xs:attribute name="PolicyIdRef" type="xs:anyURI"  
2202 use="required"/>  
2203     </xs:extension>  
2204   </xs:complexContent>  
2205 </xs:complexType>
```

2206 The <PolicyCombinerParameters> element is of PolicyCombinerParametersType complex  
2207 type.

2208 The <PolicyCombinerParameters> element contains the following attribute:

2209 PolicyIdRef [Required]

2210       The identifier of a <Policy> or the value of a <PolicyIdReference> contained in the *policy*  
2211 *set*.

2212 Support for the <PolicyCombinerParameters> element is optional, only if support for combiner  
2213 parameters is not implemented.

## 2214 **5.20 Element <PolicySetCombinerParameters>**

2215 The <PolicySetCombinerParameters> element conveys parameters associated with a particular  
2216 *policy set* within a *policy set* for a *policy-combining algorithm*.

2217 Each <PolicySetCombinerParameters> element MUST be associated with a *policy set* contained  
2218 within the same *policy set*. If multiple <PolicySetCombinerParameters> elements reference the  
2219 same *policy set*, they SHALL be considered equal to one <PolicySetCombinerParameters>  
2220 element containing the concatenation of all the sequences of <CombinerParameters> contained in all  
2221 the aforementioned <PolicySetCombinerParameters> elements, such that the order of occurrence  
2222 of the <PolicySetCombinerParameters> elements is preserved in the concatenation of the  
2223 <CombinerParameter> elements.

2224 Note that none of the *policy-combining algorithms* specified in XACML 3.0 is parameterized.

```
2225 <xs:element name="PolicySetCombinerParameters"  
2226 type="xacml:PolicySetCombinerParametersType"/>  
2227 <xs:complexType name="PolicySetCombinerParametersType">
```

```

2228     <xs:complexContent>
2229         <xs:extension base="xacml:CombinerParametersType">
2230             <xs:attribute name="PolicySetIdRef" type="xs:anyURI"
2231 use="required"/>
2232         </xs:extension>
2233     </xs:complexContent>
2234 </xs:complexType>

```

2235 The <PolicySetCombinerParameters> element is of PolicySetCombinerParametersType  
2236 complex type.

2237 The <PolicySetCombinerParameters> element contains the following attribute:

2238 PolicySetIdRef [Required]

2239 The identifier of a <PolicySet> or the value of a <PolicySetIdReference> contained in the  
2240 **policy set**.

2241 Support for the <PolicySetCombinerParameters> element is optional, only if support for combiner  
2242 parameters is not implemented.

## 2243 5.21 Element <Rule>

2244 The <Rule> element SHALL define the individual **rules** in the **policy**. The main components of this  
2245 element are the <Target>, <Condition>, <ObligationExpressions> and  
2246 <AdviceExpressions> elements and the Effect attribute.

2247 A <Rule> element may be evaluated, in which case the evaluation procedure defined in Section 7.10  
2248 SHALL be used.

```

2249 <xs:element name="Rule" type="xacml:RuleType"/>
2250 <xs:complexType name="RuleType">
2251     <xs:sequence>
2252         <xs:element ref="xacml:Description" minOccurs="0"/>
2253         <xs:element ref="xacml:Target" minOccurs="0"/>
2254         <xs:element ref="xacml:Condition" minOccurs="0"/>
2255         <xs:element ref="xacml:ObligationExpressions" minOccurs="0"/>
2256         <xs:element ref="xacml:AdviceExpressions" minOccurs="0"/>
2257     </xs:sequence>
2258     <xs:attribute name="RuleId" type="xs:string" use="required"/>
2259     <xs:attribute name="Effect" type="xacml:EffectType" use="required"/>
2260 </xs:complexType>

```

2261 The <Rule> element is of RuleType complex type.

2262 The <Rule> element contains the following attributes and elements:

2263 RuleId [Required]

2264 A string identifying this **rule**.

2265 Effect [Required]

2266 **Rule effect**. The value of this attribute is either "Permit" or "Deny".

2267 <Description> [Optional]

2268 A free-form description of the **rule**.

2269 <Target> [Optional]

2270 Identifies the set of **decision requests** that the <Rule> element is intended to evaluate. If this  
2271 element is omitted, then the **target** for the <Rule> SHALL be defined by the <Target> element  
2272 of the enclosing <Policy> element. See Section 7.7 for details.

2273 <Condition> [Optional]

2274 A **predicate** that MUST be satisfied for the **rule** to be assigned its Effect value.



2275 <ObligationExpressions> [Optional]

2276 A **conjunctive sequence** of **obligation** expressions which MUST be evaluated into **obligations**  
2277 by the PDP. The corresponding **obligations** MUST be fulfilled by the **PEP** in conjunction with the  
2278 **authorization decision**. See Section 7.18 for a description of how the set of **obligations** to be  
2279 returned by the **PDP** SHALL be determined. See section 7.2 about enforcement of **obligations**.

2280 <AdviceExpressions> [Optional]

2281 A **conjunctive sequence** of **advice** expressions which MUST be evaluated into **advice** by the **PDP**.  
2282 The corresponding **advice** provide supplementary information to the **PEP** in conjunction with the  
2283 **authorization decision**. See Section 7.18 for a description of how the set of **advice** to be  
2284 returned by the **PDP** SHALL be determined.

## 2285 5.22 Simple type EffectType

2286 The EffectType simple type defines the values allowed for the Effect attribute of the <Rule> element  
2287 and for the FulfillOn attribute of the <ObligationExpression> and <AdviceExpression>  
2288 elements.

```
2289 <xs:simpleType name="EffectType">  
2290   <xs:restriction base="xs:string">  
2291     <xs:enumeration value="Permit"/>  
2292     <xs:enumeration value="Deny"/>  
2293   </xs:restriction>  
2294 </xs:simpleType>
```

## 2295 5.23 Element <VariableDefinition>

2296 The <VariableDefinition> element SHALL be used to define a value that can be referenced by a  
2297 <VariableReference> element. The name supplied for its VariableId attribute SHALL NOT occur  
2298 in the VariableId attribute of any other <VariableDefinition> element within the encompassing  
2299 **policy**. The <VariableDefinition> element MAY contain undefined <VariableReference>  
2300 elements, but if it does, a corresponding <VariableDefinition> element MUST be defined later in  
2301 the encompassing **policy**. <VariableDefinition> elements MAY be grouped together or MAY be  
2302 placed close to the reference in the encompassing **policy**. There MAY be zero or more references to  
2303 each <VariableDefinition> element.

```
2304 <xs:element name="VariableDefinition" type="xacml:VariableDefinitionType"/>  
2305 <xs:complexType name="VariableDefinitionType">  
2306   <xs:sequence>  
2307     <xs:element ref="xacml:Expression"/>  
2308   </xs:sequence>  
2309   <xs:attribute name="VariableId" type="xs:string" use="required"/>  
2310 </xs:complexType>
```

2311 The <VariableDefinition> element is of VariableDefinitionType complex type. The  
2312 <VariableDefinition> element has the following elements and attributes:

2313 <Expression> [Required]

2314 Any element of ExpressionType complex type.

2315 VariableId [Required]

2316 The name of the variable definition.

## 2317 5.24 Element <VariableReference>

2318 The <VariableReference> element is used to reference a value defined within the same  
2319 encompassing <Policy> element. The <VariableReference> element SHALL refer to the  
2320 <VariableDefinition> element by **identifier equality** on the value of their respective VariableId



2321 attributes. One and only one <VariableDefinition> MUST exist within the same encompassing  
2322 <Policy> element to which the <VariableReference> refers. There MAY be zero or more  
2323 <VariableReference> elements that refer to the same <VariableDefinition> element.

```
2324 <xs:element name="VariableReference" type="xacml:VariableReferenceType"  
2325 substitutionGroup="xacml:Expression"/>  
2326 <xs:complexType name="VariableReferenceType">  
2327 <xs:complexContent>  
2328 <xs:extension base="xacml:ExpressionType">  
2329 <xs:attribute name="VariableId" type="xs:string"  
2330 use="required"/>  
2331 </xs:extension>  
2332 </xs:complexContent>  
2333 </xs:complexType>
```

2334 The <VariableReference> element is of the VariableReferenceType complex type, which is of  
2335 the ExpressionType complex type and is a member of the <Expression> element substitution group.  
2336 The <VariableReference> element MAY appear any place where an <Expression> element occurs  
2337 in the schema.

2338 The <VariableReference> element has the following attribute:

2339 VariableId [Required]

2340 The name used to refer to the value defined in a <VariableDefinition> element.

## 2341 5.25 Element <Expression>

2342 The <Expression> element is not used directly in a **policy**. The <Expression> element signifies that  
2343 an element that extends the ExpressionType and is a member of the <Expression> element  
2344 substitution group SHALL appear in its place.

```
2345 <xs:element name="Expression" type="xacml:ExpressionType" abstract="true"/>  
2346 <xs:complexType name="ExpressionType" abstract="true"/>
```

2347 The following elements are in the <Expression> element substitution group:

2348 <Apply>, <AttributeSelector>, <AttributeValue>, <Function>, <VariableReference> and  
2349 <AttributeDesignator>.

## 2350 5.26 Element <Condition>

2351 The <Condition> element is a Boolean function over **attributes** or functions of **attributes**.

```
2352 <xs:element name="Condition" type="xacml:ConditionType"/>  
2353 <xs:complexType name="ConditionType">  
2354 <xs:sequence>  
2355 <xs:element ref="xacml:Expression"/>  
2356 </xs:sequence>  
2357 </xs:complexType>
```

2358 The <Condition> contains one <Expression> element, with the restriction that the <Expression>  
2359 return data-type MUST be "http://www.w3.org/2001/XMLSchema#boolean". Evaluation of the  
2360 <Condition> element is described in Section 7.9.

## 2361 5.27 Element <Apply>

2362 The <Apply> element denotes application of a function to its arguments, thus encoding a function call.  
2363 The <Apply> element can be applied to any combination of the members of the <Expression>  
2364 element substitution group. See Section 5.25.

```
2365 <xs:element name="Apply" type="xacml:ApplyType"  
2366 substitutionGroup="xacml:Expression"/>
```

```

2367 <xs:complexType name="ApplyType">
2368   <xs:complexContent>
2369     <xs:extension base="xacml:ExpressionType">
2370       <xs:sequence>
2371         <xs:element ref="xacml:Description" minOccurs="0"/>
2372         <xs:element ref="xacml:Expression" minOccurs="0"
2373           maxOccurs="unbounded"/>
2374       </xs:sequence>
2375       <xs:attribute name="FunctionId" type="xs:anyURI"
2376         use="required"/>
2377     </xs:extension>
2378   </xs:complexContent>
2379 </xs:complexType>

```

2380 The <Apply> element is of ApplyType complex type.

2381 The <Apply> element contains the following attributes and elements:

2382 FunctionId [Required]

2383       The identifier of the function to be applied to the arguments. XACML-defined functions are  
2384       described in Appendix A.3.

2385 <Description> [Optional]

2386       A free-form description of the <Apply> element.

2387 <Expression> [Optional]

2388       Arguments to the function, which may include other functions.

## 2389 5.28 Element <Function>

2390 The <Function> element SHALL be used to name a function as an argument to the function defined by  
2391 the parent <Apply> element.

```

2392 <xs:element name="Function" type="xacml:FunctionType"
2393 substitutionGroup="xacml:Expression"/>
2394 <xs:complexType name="FunctionType">
2395   <xs:complexContent>
2396     <xs:extension base="xacml:ExpressionType">
2397       <xs:attribute name="FunctionId" type="xs:anyURI"
2398         use="required"/>
2399     </xs:extension>
2400   </xs:complexContent>
2401 </xs:complexType>

```

2402 The <Function> element is of FunctionType complex type.

2403 The <Function> element contains the following attribute:

2404 FunctionId [Required]

2405       The identifier of the function.

## 2406 5.29 Element <AttributeDesignator>

2407 The <AttributeDesignator> element retrieves a **bag** of values for a **named attribute** from the  
2408 request **context**. A **named attribute** SHALL be considered present if there is at least one **attribute** that  
2409 matches the criteria set out below.

2410 The <AttributeDesignator> element SHALL return a **bag** containing all the **attribute** values that are  
2411 matched by the **named attribute**. In the event that no matching **attribute** is present in the **context**, the  
2412 MustBePresent attribute governs whether this element returns an empty **bag** or "Indeterminate". See  
2413 Section 7.3.5.

2414 The <AttributeDesignator> MAY appear in the <Match> element and MAY be passed to the  
2415 <Apply> element as an argument.

2416 The <AttributeDesignator> element is of the AttributeDesignatorType complex type.

```
2417 <xs:element name="AttributeDesignator" type="xacml:AttributeDesignatorType"  
2418 substitutionGroup="xacml:Expression"/>  
2419 <xs:complexType name="AttributeDesignatorType">  
2420 <xs:complexContent>  
2421 <xs:extension base="xacml:ExpressionType">  
2422 <xs:attribute name="Category" type="xs:anyURI"  
2423 use="required"/>  
2424 <xs:attribute name="AttributeId" type="xs:anyURI"  
2425 use="required"/>  
2426 <xs:attribute name="DataType" type="xs:anyURI"  
2427 use="required"/>  
2428 <xs:attribute name="Issuer" type="xs:string" use="optional"/>  
2429 <xs:attribute name="MustBePresent" type="xs:boolean"  
2430 use="required"/>  
2431 </xs:extension>  
2432 </xs:complexContent>  
2433 </xs:complexType>
```

2434 A **named attribute** SHALL match an **attribute** if the values of their respective Category,  
2435 AttributeId, DataType and Issuer attributes match. The attribute designator's Category MUST  
2436 match, by **identifier equality**, the Category of the <Attributes> element in which the **attribute** is  
2437 present. The attribute designator's AttributeId MUST match, by **identifier equality**, the  
2438 AttributeId of the attribute. The attribute designator's DataType MUST match, by **identifier**  
2439 **equality**, the DataType of the same **attribute**.

2440 If the Issuer attribute is present in the attribute designator, then it MUST match, using the  
2441 "urn:oasis:names:tc:xacml:1.0:function:string-equal" function, the Issuer of the same **attribute**. If the  
2442 Issuer is not present in the attribute designator, then the matching of the **attribute** to the **named**  
2443 **attribute** SHALL be governed by AttributeId and DataType attributes alone.

2444 The <AttributeDesignatorType> contains the following attributes:

2445 Category [Required]

2446 This attribute SHALL specify the Category with which to match the **attribute**.

2447 AttributeId [Required]

2448 This attribute SHALL specify the AttributeId with which to match the **attribute**.

2449 DataType [Required]

2450 The **bag** returned by the <AttributeDesignator> element SHALL contain values of this data-  
2451 type.

2452 Issuer [Optional]

2453 This attribute, if supplied, SHALL specify the Issuer with which to match the **attribute**.

2454 MustBePresent [Required]

2455 This attribute governs whether the element returns "Indeterminate" or an empty **bag** in the event  
2456 the **named attribute** is absent from the request **context**. See Section 7.3.5. Also see Sections  
2457 7.19.2 and 7.19.3.

## 2458 5.30 Element <AttributeSelector>

2459 The <AttributeSelector> element produces a **bag** of unnamed and uncategorized **attribute** values.  
2460 The values shall be constructed from the node(s) selected by applying the XPath expression given by the  
2461 element's Path attribute to the XML content indicated by the element's Category attribute. Support for  
2462 the <AttributeSelector> element is OPTIONAL.

2463 See section 7.3.7 for details of <AttributeSelector> evaluation.

```
2464 <xs:element name="AttributeSelector" type="xacml:AttributeSelectorType"  
2465 substitutionGroup="xacml:Expression"/>  
2466 <xs:complexType name="AttributeSelectorType">  
2467   <xs:complexContent>  
2468     <xs:extension base="xacml:ExpressionType">  
2469       <xs:attribute name="Category" type="xs:anyURI"  
2470         use="required"/>  
2471       <xs:attribute name="ContextSelectorId" type="xs:anyURI"  
2472         use="optional"/>  
2473       <xs:attribute name="Path" type="xs:string"  
2474         use="required"/>  
2475       <xs:attribute name="DataType" type="xs:anyURI"  
2476         use="required"/>  
2477       <xs:attribute name="MustBePresent" type="xs:boolean"  
2478         use="required"/>  
2479     </xs:extension>  
2480   </xs:complexContent>  
2481 </xs:complexType>
```

2482 The <AttributeSelector> element is of AttributeSelectorType complex type.

2483 The <AttributeSelector> element has the following attributes:

2484 Category [Required]

2485 This attribute SHALL specify the **attributes** category of the <Content> element containing the  
2486 XML from which nodes will be selected. It also indicates the **attributes** category containing the  
2487 applicable ContextSelectorId attribute, if the element includes a ContextSelectorId xml  
2488 attribute.

2489 ContextSelectorId [Optional]

2490 This attribute refers to the **attribute** (by its AttributeId) in the request **context** in the category  
2491 given by the Category attribute. The referenced **attribute** MUST have data type  
2492 urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression, and must select a single node in the  
2493 <Content> element. The XPathCategory attribute of the referenced **attribute** MUST be equal  
2494 to the Category attribute of the **attribute selector**.

2495 Path [Required]

2496 This attribute SHALL contain an XPath expression to be evaluated against the specified XML  
2497 content. See Section 7.3.7 for details of the XPath evaluation during <AttributeSelector>  
2498 processing. The namespace context for the value of the Path attribute is given by the [in-scope  
2499 namespaces] [INFOSET] of the <AttributeSelector> element.

2500 DataType [Required]

2501 The attribute specifies the datatype of the values returned from the evaluation of this  
2502 <AttributeSelector> element.

2503 MustBePresent [Required]

2504 This attribute governs whether the element returns "Indeterminate" or an empty **bag** in the event  
2505 that the attributes category specified by the Category attribute does not exist in the request  
2506 context, or the attributes category does exist but it does not have a <Content> child element, or  
2507 the <Content> element does exist but the XPath expression selects no node. See Section 7.3.5.  
2508 Also see Sections 7.19.2 and 7.19.3.

## 2509 5.31 Element <AttributeValue>

2510 The <AttributeValue> element SHALL contain a literal **attribute** value.

```

2511 <xs:element name="AttributeValue" type="xacml:AttributeValueType"
2512 substitutionGroup="xacml:Expression"/>
2513 <xs:complexType name="AttributeValueType" mixed="true">
2514   <xs:complexContent mixed="true">
2515     <xs:extension base="xacml:ExpressionType">
2516       <xs:sequence>
2517         <xs:any namespace="##any" processContents="lax"
2518           minOccurs="0" maxOccurs="unbounded"/>
2519       </xs:sequence>
2520       <xs:attribute name="DataType" type="xs:anyURI"
2521         use="required"/>
2522       <xs:anyAttribute namespace="##any" processContents="lax"/>
2523     </xs:extension>
2524   </xs:complexContent>
2525 </xs:complexType>

```

2526 The <AttributeValue> element is of AttributeValueType complex type.

2527 The <AttributeValue> element has the following attributes:

2528 DataType [Required]

2529 The data-type of the *attribute* value.

### 2530 5.32 Element <Obligations>

2531 The <Obligations> element SHALL contain a set of <Obligation> elements.

```

2532 <xs:element name="Obligations" type="xacml:ObligationsType"/>
2533 <xs:complexType name="ObligationsType">
2534   <xs:sequence>
2535     <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>
2536   </xs:sequence>
2537 </xs:complexType>

```

2538 The <Obligations> element is of ObligationsType complexType.

2539 The <Obligations> element contains the following element:

2540 <Obligation> [One to Many]

2541 A sequence of *obligations*. See Section 5.34.

### 2542 5.33 Element <AssociatedAdvice>

2543 The <AssociatedAdvice> element SHALL contain a set of <Advice> elements.

```

2544 <xs:element name="AssociatedAdvice" type="xacml:AssociatedAdviceType"/>
2545 <xs:complexType name="AssociatedAdviceType">
2546   <xs:sequence>
2547     <xs:element ref="xacml:Advice" maxOccurs="unbounded"/>
2548   </xs:sequence>
2549 </xs:complexType>

```

2550 The <AssociatedAdvice> element is of AssociatedAdviceType complexType.

2551 The <AssociatedAdvice> element contains the following element:

2552 <Advice> [One to Many]

2553 A sequence of *advice*. See Section 5.35.

### 2554 5.34 Element <Obligation>

2555 The <Obligation> element SHALL contain an identifier for the *obligation* and a set of *attributes* that  
2556 form arguments of the action defined by the *obligation*.

```

2557 <xs:element name="Obligation" type="xacml:ObligationType"/>
2558 <xs:complexType name="ObligationType">
2559   <xs:sequence>
2560     <xs:element ref="xacml:AttributeAssignment" minOccurs="0"
2561       maxOccurs="unbounded"/>
2562   </xs:sequence>
2563   <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
2564 </xs:complexType>

```

2565 The <Obligation> element is of ObligationType complexType. See Section 7.18 for a description  
 2566 of how the set of **obligations** to be returned by the **PDP** is determined.

2567 The <Obligation> element contains the following elements and attributes:

2568 ObligationId [Required]

2569 **Obligation** identifier. The value of the **obligation** identifier SHALL be interpreted by the **PEP**.

2570 <AttributeAssignment> [Optional]

2571 **Obligation** arguments assignment. The values of the **obligation** arguments SHALL be  
 2572 interpreted by the **PEP**.

### 2573 5.35 Element <Advice>

2574 The <Advice> element SHALL contain an identifier for the **advice** and a set of **attributes** that form  
 2575 arguments of the supplemental information defined by the **advice**.

```

2576 <xs:element name="Advice" type="xacml:AdviceType"/>
2577 <xs:complexType name="AdviceType">
2578   <xs:sequence>
2579     <xs:element ref="xacml:AttributeAssignment" minOccurs="0"
2580       maxOccurs="unbounded"/>
2581   </xs:sequence>
2582   <xs:attribute name="AdviceId" type="xs:anyURI" use="required"/>
2583 </xs:complexType>

```

2584 The <Advice> element is of AdviceType complexType. See Section 7.18 for a description of how the  
 2585 set of **advice** to be returned by the **PDP** is determined.

2586 The <Advice> element contains the following elements and attributes:

2587 AdviceId [Required]

2588 **Advice** identifier. The value of the **advice** identifier MAY be interpreted by the **PEP**.

2589 <AttributeAssignment> [Optional]

2590 **Advice** arguments assignment. The values of the **advice** arguments MAY be interpreted by the  
 2591 **PEP**.

### 2592 5.36 Element <AttributeAssignment>

2593 The <AttributeAssignment> element is used for including arguments in **obligation** and **advice**  
 2594 expressions. It SHALL contain an AttributeId and the corresponding **attribute** value, by extending  
 2595 the AttributeValueType type definition. The <AttributeAssignment> element MAY be used in  
 2596 any way that is consistent with the schema syntax, which is a sequence of <xs:any> elements. The  
 2597 value specified SHALL be understood by the **PEP**, but it is not further specified by XACML. See Section  
 2598 7.18. Section 4.2.4.3 provides a number of examples of arguments included in **obligation** expressions.

```

2599 <xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
2600 <xs:complexType name="AttributeAssignmentType" mixed="true">
2601   <xs:complexContent>
2602     <xs:extension base="xacml:AttributeValueType">
2603       <xs:attribute name="AttributeId" type="xs:anyURI"
2604         use="required"/>

```



```
2605         <xs:attribute name="Category" type="xs:anyURI"
2606             use="optional" />
2607         <xs:attribute name="Issuer" type="xs:string" use="optional" />
2608     </xs:extension>
2609 </xs:complexContent>
2610 </xs:complexType>
```

2611 The <AttributeAssignment> element is of AttributeAssignmentType complex type.

2612 The <AttributeAssignment> element contains the following attributes:

2613 AttributeId [Required]

2614 The **attribute** Identifier.

2615 Category [Optional]

2616 An optional category of the **attribute**. If this attribute is missing, the **attribute** has no category.

2617 The **PEP** SHALL interpret the significance and meaning of any Category attribute. Non-

2618 normative note: an expected use of the category is to disambiguate **attributes** which are relayed

2619 from the request.

2620 Issuer [Optional]

2621 An optional issuer of the **attribute**. If this attribute is missing, the **attribute** has no issuer. The

2622 **PEP** SHALL interpret the significance and meaning of any Issuer attribute. Non-normative note:

2623 an expected use of the issuer is to disambiguate **attributes** which are relayed from the request.

## 2624 5.37 Element <ObligationExpressions>

2625 The <ObligationExpressions> element SHALL contain a set of <ObligationExpression>

2626 elements.

```
2627 <xs:element name="ObligationExpressions"
2628     type="xacml:ObligationExpressionsType" />
2629 <xs:complexType name="ObligationExpressionsType">
2630     <xs:sequence>
2631         <xs:element ref="xacml:ObligationExpression" maxOccurs="unbounded" />
2632     </xs:sequence>
2633 </xs:complexType>
```

2634 The <ObligationExpressions> element is of ObligationExpressionsType complexType.

2635 The <ObligationExpressions> element contains the following element:

2636 <ObligationExpression> [One to Many]

2637 A sequence of **obligations** expressions. See Section 5.39.

## 2638 5.38 Element <AdviceExpressions>

2639 The <AdviceExpressions> element SHALL contain a set of <AdviceExpression> elements.

```
2640 <xs:element name="AdviceExpressions" type="xacml:AdviceExpressionsType" />
2641 <xs:complexType name="AdviceExpressionsType">
2642     <xs:sequence>
2643         <xs:element ref="xacml:AdviceExpression" maxOccurs="unbounded" />
2644     </xs:sequence>
2645 </xs:complexType>
```

2646 The <AdviceExpressions> element is of AdviceExpressionsType complexType.

2647 The <AdviceExpressions> element contains the following element:

2648 <AdviceExpression> [One to Many]

2649 A sequence of **advice** expressions. See Section 5.40.



### 2650 5.39 Element <ObligationExpression>

2651 The <ObligationExpression> element evaluates to an **obligation** and SHALL contain an identifier  
2652 for an **obligation** and a set of expressions that form arguments of the action defined by the **obligation**.  
2653 The FulfillOn attribute SHALL indicate the **effect** for which this **obligation** must be fulfilled by the  
2654 **PEP**.

```
2655 <xs:element name="ObligationExpression"  
2656     type="xacml:ObligationExpressionType"/>  
2657 <xs:complexType name="ObligationExpressionType">  
2658   <xs:sequence>  
2659     <xs:element ref="xacml:AttributeAssignmentExpression" minOccurs="0"  
2660       maxOccurs="unbounded"/>  
2661   </xs:sequence>  
2662   <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>  
2663   <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>  
2664 </xs:complexType>
```

2665 The <ObligationExpression> element is of ObligationExpressionType complexType. See  
2666 Section 7.18 for a description of how the set of **obligations** to be returned by the **PDP** is determined.

2667 The <ObligationExpression> element contains the following elements and attributes:

2668 ObligationId [Required]

2669 **Obligation** identifier. The value of the **obligation** identifier SHALL be interpreted by the **PEP**.

2670 FulfillOn [Required]

2671 The **effect** for which this **obligation** must be fulfilled by the **PEP**.

2672 <AttributeAssignmentExpression> [Optional]

2673 **Obligation** arguments in the form of expressions. The expressions SHALL be evaluated by the  
2674 PDP to constant <AttributeValue> elements or **bags**, which shall be the attribute  
2675 assignments in the <Obligation> returned to the PEP. If an  
2676 <AttributeAssignmentExpression> evaluates to an atomic **attribute** value, then there  
2677 MUST be one resulting <AttributeAssignment> which MUST contain this single **attribute**  
2678 value. If the <AttributeAssignmentExpression> evaluates to a **bag**, then there MUST be a  
2679 resulting <AttributeAssignment> for each of the values in the **bag**. If the **bag** is empty, there  
2680 shall be no <AttributeAssignment> from this <AttributeAssignmentExpression>. The  
2681 values of the **obligation** arguments SHALL be interpreted by the **PEP**.

### 2682 5.40 Element <AdviceExpression>

2683 The <AdviceExpression> element evaluates to an **advice** and SHALL contain an identifier for an  
2684 **advice** and a set of expressions that form arguments of the supplemental information defined by the  
2685 **advice**. The AppliesTo attribute SHALL indicate the **effect** for which this **advice** must be provided to  
2686 the **PEP**.

```
2687 <xs:element name="AdviceExpression" type="xacml:AdviceExpressionType"/>  
2688 <xs:complexType name="AdviceExpressionType">  
2689   <xs:sequence>  
2690     <xs:element ref="xacml:AttributeAssignmentExpression" minOccurs="0"  
2691       maxOccurs="unbounded"/>  
2692   </xs:sequence>  
2693   <xs:attribute name="AdviceId" type="xs:anyURI" use="required"/>  
2694   <xs:attribute name="AppliesTo" type="xacml:EffectType" use="required"/>  
2695 </xs:complexType>
```

2696 The <AdviceExpression> element is of AdviceExpressionType complexType. See Section 7.18  
2697 for a description of how the set of **advice** to be returned by the **PDP** is determined.

2698 The <AdviceExpression> element contains the following elements and attributes:

2699 AdviceId [Required]  
 2700         **Advice** identifier. The value of the **advice** identifier MAY be interpreted by the **PEP**.  
 2701 AppliesTo [Required]  
 2702         The **effect** for which this **advice** must be provided to the **PEP**.  
 2703 <AttributeAssignmentExpression> [Optional]  
 2704         **Advice** arguments in the form of expressions. The expressions SHALL be evaluated by the PDP  
 2705 to constant <AttributeValue> elements or **bags**, which shall be the attribute assignments in  
 2706 the <Advice> returned to the PEP. If an <AttributeAssignmentExpression> evaluates to  
 2707 an atomic **attribute** value, then there MUST be one resulting <AttributeAssignment> which  
 2708 MUST contain this single **attribute** value. If the <AttributeAssignmentExpression>  
 2709 evaluates to a **bag**, then there MUST be a resulting <AttributeAssignment> for each of the  
 2710 values in the **bag**. If the **bag** is empty, there shall be no <AttributeAssignment> from this  
 2711 <AttributeAssignmentExpression>. The values of the **advice** arguments MAY be  
 2712 interpreted by the **PEP**.

## 2713 5.41 Element <AttributeAssignmentExpression>

2714 The <AttributeAssignmentExpression> element is used for including arguments in **obligations**  
 2715 and **advice**. It SHALL contain an AttributeId and an expression which SHALL be evaluated into the  
 2716 corresponding **attribute** value. The value specified SHALL be understood by the **PEP**, but it is not further  
 2717 specified by XACML. See Section 7.18. Section 4.2.4.3 provides a number of examples of arguments  
 2718 included in **obligations**.

```

2719 <xs:element name="AttributeAssignmentExpression"
2720    type="xacml:AttributeAssignmentExpressionType"/>
2721 <xs:complexType name="AttributeAssignmentExpressionType">
2722    <xs:sequence>
2723     <xs:element ref="xacml:Expression"/>
2724    </xs:sequence>
2725    <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
2726    <xs:attribute name="Category" type="xs:anyURI" use="optional"/>
2727    <xs:attribute name="Issuer" type="xs:string" use="optional"/>
2728 </xs:complexType>
  
```

2729 The <AttributeAssignmentExpression> element is of AttributeAssignmentExpressionType  
 2730 complex type.

2731 The <AttributeAssignmentExpression> element contains the following attributes:

2732 <Expression> [Required]

2733         The expression which evaluates to a constant **attribute** value or a bag of zero or more attribute  
 2734 values. See section 5.25.

2735 AttributeId [Required]

2736         The **attribute** identifier. The value of the AttributeId attribute in the resulting  
 2737 <AttributeAssignment> element MUST be equal to this value.

2738 Category [Optional]

2739         An optional category of the **attribute**. If this attribute is missing, the **attribute** has no category.  
 2740 The value of the Category attribute in the resulting <AttributeAssignment> element MUST be  
 2741 equal to this value.

2742 Issuer [Optional]

2743         An optional issuer of the **attribute**. If this attribute is missing, the **attribute** has no issuer. The  
 2744 value of the Issuer attribute in the resulting <AttributeAssignment> element MUST be equal to  
 2745 this value.

## 2746 5.42 Element <Request>

2747 The <Request> element is an abstraction layer used by the **policy** language. For simplicity of  
2748 expression, this document describes **policy** evaluation in terms of operations on the **context**. However a  
2749 conforming **PDP** is not required to actually instantiate the **context** in the form of an XML document. But,  
2750 any system conforming to the XACML specification MUST produce exactly the same **authorization**  
2751 **decisions** as if all the inputs had been transformed into the form of an <Request> element.

```
2752 <xs:element name="Request" type="xacml:RequestType"/>
2753 <xs:complexType name="RequestType">
2754   <xs:sequence>
2755     <xs:element ref="xacml:RequestDefaults" minOccurs="0"/>
2756     <xs:element ref="xacml:Attributes" maxOccurs="unbounded"/>
2757     <xs:element ref="xacml:MultiRequests" minOccurs="0"/>
2758   </xs:sequence>
2759   <xs:attribute name="ReturnPolicyIdList" type="xs:boolean" use="required"/>
2760   <xs:attribute name="CombinedDecision" type="xs:boolean" use="required" />
2761 </xs:complexType>
```

2762 The <Request> element is of RequestType complex type.

2763 The <Request> element contains the following elements and attributes:

2764 ReturnPolicyIdList [Required]

2765 This attribute is used to request that the **PDP** return a list of all fully applicable **policies** and  
2766 **policy sets** which were used in the decision as a part of the decision response.

2767 CombinedDecision [Required]

2768 This attribute is used to request that the **PDP** combines multiple decisions into a single decision.  
2769 The use of this attribute is specified in [Multi]. If the **PDP** does not implement the relevant  
2770 functionality in [Multi], then the **PDP** must return an Indeterminate with a status code of  
2771 urn:oasis:names:tc:xacml:1.0:status:processing-error if it receives a request with this attribute set  
2772 to "true".

2773 <RequestDefaults> [Optional]

2774 Contains default values for the request, such as XPath version. See section 5.43.

2775 <Attributes> [One to Many]

2776 Specifies information about **attributes** of the request **context** by listing a sequence of  
2777 <Attribute> elements associated with an **attribute** category. One or more <Attributes>  
2778 elements are allowed. Different <Attributes> elements with different categories are used to  
2779 represent information about the **subject**, **resource**, **action**, **environment** or other categories of  
2780 the **access** request.

2781 The <Request> element contains <Attributes> elements. There may be multiple  
2782 <Attributes> elements with the same Category attribute if the **PDP** implements the multiple  
2783 decision profile, see [Multi]. Under other conditions, it is a syntax error if there are multiple  
2784 <Attributes> elements with the same Category (see Section 7.19.2 for error codes).

2785 <MultiRequests> [Optional]

2786 Lists multiple **request contexts** by references to the <Attributes> elements. Implementation  
2787 of this element is optional. The semantics of this element is defined in [Multi]. If the  
2788 implementation does not implement this element, it MUST return an Indeterminate result if it  
2789 encounters this element. See section 5.50.

## 2790 5.43 Element <RequestDefaults>

2791 The <RequestDefaults> element SHALL specify default values that apply to the <Request> element.

```
2792 <xs:element name="RequestDefaults" type="xacml:RequestDefaultsType"/>
```

```

2793 <xs:complexType name="RequestDefaultsType">
2794   <xs:sequence>
2795     <xs:choice>
2796       <xs:element ref="xacml:XPathVersion"/>
2797     </xs:choice>
2798   </xs:sequence>
2799 </xs:complexType>

```

2800 <RequestDefaults> element is of RequestDefaultsType complex type.

2801 The <RequestDefaults> element contains the following elements:

2802 <XPathVersion> [Optional]

2803 Default XPath version for XPath expressions occurring in the request.

## 2804 5.44 Element <Attributes>

2805 The <Attributes> element specifies **attributes** of a **subject**, **resource**, **action**, **environment** or  
 2806 another category by listing a sequence of <Attribute> elements associated with the category.

```

2807 <xs:element name="Attributes" type="xacml:AttributesType"/>
2808 <xs:complexType name="AttributesType">
2809   <xs:sequence>
2810     <xs:element ref="xacml:Content" minOccurs="0"/>
2811     <xs:element ref="xacml:Attribute" minOccurs="0"
2812       maxOccurs="unbounded"/>
2813   </xs:sequence>
2814   <xs:attribute name="Category" type="xs:anyURI" use="required"/>
2815   <xs:attribute ref="xml:id" use="optional"/>
2816 </xs:complexType>

```

2817 The <Attributes> element is of AttributesType complex type.

2818 The <Attributes> element contains the following elements and attributes:

2819 Category [Required]

2820 This attribute indicates which **attribute** category the contained **attributes** belong to. The  
 2821 Category attribute is used to differentiate between **attributes** of **subject**, **resource**, **action**,  
 2822 **environment** or other categories.

2823 xml:id [Optional]

2824 This attribute provides a unique identifier for this <Attributes> element. See [XMLid] It is  
 2825 primarily intended to be referenced in multiple requests. See [Multi].

2826 <Content> [Optional]

2827 Specifies additional sources of **attributes** in free form XML document format which can be  
 2828 referenced using <AttributeSelector> elements.

2829 <Attribute> [Any Number]

2830 A sequence of **attributes** that apply to the category of the request.

## 2831 5.45 Element <Content>

2832 The <Content> element is a notional placeholder for additional **attributes**, typically the content of the  
 2833 **resource**.

```

2834 <xs:element name="Content" type="xacml:ContentType"/>
2835 <xs:complexType name="ContentType" mixed="true">
2836   <xs:sequence>
2837     <xs:any namespace="##any" processContents="lax"/>
2838   </xs:sequence>
2839 </xs:complexType>

```

2840 The <Content> element is of Contentype complex type.  
2841 The <Content> element has exactly one arbitrary type child element.

## 2842 5.46 Element <Attribute>

2843 The <Attribute> element is the central abstraction of the request **context**. It contains **attribute** meta-  
2844 data and one or more **attribute** values. The **attribute** meta-data comprises the **attribute** identifier and  
2845 the **attribute** issuer. <AttributeDesignator> elements in the **policy** MAY refer to **attributes** by  
2846 means of this meta-data.

```
2847 <xs:element name="Attribute" type="xacml:AttributeType"/>  
2848 <xs:complexType name="AttributeType">  
2849   <xs:sequence>  
2850     <xs:element ref="xacml:AttributeValue" maxOccurs="unbounded"/>  
2851   </xs:sequence>  
2852   <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>  
2853   <xs:attribute name="Issuer" type="xs:string" use="optional"/>  
2854   <xs:attribute name="IncludeInResult" type="xs:boolean" use="required"/>  
2855 </xs:complexType>
```

2856 The <Attribute> element is of AttributeType complex type.

2857 The <Attribute> element contains the following attributes and elements:

2858 AttributeId [Required]

2859       The **Attribute** identifier. A number of identifiers are reserved by XACML to denote commonly  
2860 used **attributes**. See Appendix Appendix B.

2861 Issuer [Optional]

2862       The **Attribute** issuer. For example, this attribute value MAY be an x500Name that binds to a  
2863 public key, or it may be some other identifier exchanged out-of-band by issuing and relying  
2864 parties.

2865 IncludeInResult [Default: false]

2866       Whether to include this **attribute** in the result. This is useful to correlate requests with their  
2867 responses in case of multiple requests.

2868 <AttributeValue> [One to Many]

2869       One or more **attribute** values. Each **attribute** value MAY have contents that are empty, occur  
2870 once or occur multiple times.

## 2871 5.47 Element <Response>

2872 The <Response> element is an abstraction layer used by the **policy** language. Any proprietary system  
2873 using the XACML specification MUST transform an XACML **context** <Response> element into the form  
2874 of its **authorization decision**.

2875 The <Response> element encapsulates the **authorization decision** produced by the **PDP**. It includes a  
2876 sequence of one or more results, with one <Result> element per requested **resource**. Multiple results  
2877 MAY be returned by some implementations, in particular those that support the XACML Profile for  
2878 Requests for Multiple Resources [**Multi**]. Support for multiple results is OPTIONAL.

```
2879 <xs:element name="Response" type="xacml:ResponseType"/>  
2880 <xs:complexType name="ResponseType">  
2881   <xs:sequence>  
2882     <xs:element ref="xacml:Result" maxOccurs="unbounded"/>  
2883   </xs:sequence>  
2884 </xs:complexType>
```

2885 The <Response> element is of ResponseType complex type.

2886 The <Response> element contains the following elements:

2887 <Result> [One to Many]

2888 An **authorization decision** result. See Section 5.48.

## 2889 5.48 Element <Result>

2890 The <Result> element represents an **authorization decision** result. It MAY include a set of  
2891 **obligations** that MUST be fulfilled by the **PEP**. If the **PEP** does not understand or cannot fulfill an  
2892 **obligation**, then the action of the PEP is determined by its bias, see section 7.1. It MAY include a set of  
2893 **advice** with supplemental information which MAY be safely ignored by the **PEP**.

```
2894 <xs:complexType name="ResultType">  
2895   <xs:sequence>  
2896     <xs:element ref="xacml:Decision"/>  
2897     <xs:element ref="xacml:Status" minOccurs="0"/>  
2898     <xs:element ref="xacml:Obligations" minOccurs="0"/>  
2899     <xs:element ref="xacml:AssociatedAdvice" minOccurs="0"/>  
2900     <xs:element ref="xacml:Attributes" minOccurs="0"  
2901       maxOccurs="unbounded"/>  
2902     <xs:element ref="xacml:PolicyIdentifierList" minOccurs="0"/>  
2903   </xs:sequence>  
2904 </xs:complexType>
```

2905 The <Result> element is of ResultType complex type.

2906 The <Result> element contains the following attributes and elements:

2907 <Decision> [Required]

2908 The **authorization decision**: "Permit", "Deny", "Indeterminate" or "NotApplicable".

2909 <Status> [Optional]

2910 Indicates whether errors occurred during evaluation of the **decision request**, and optionally,  
2911 information about those errors. If the <Response> element contains <Result> elements whose  
2912 <Status> elements are all identical, and the <Response> element is contained in a protocol  
2913 wrapper that can convey status information, then the common status information MAY be placed  
2914 in the protocol wrapper and this <Status> element MAY be omitted from all <Result>  
2915 elements.

2916 <Obligations> [Optional]

2917 A list of **obligations** that MUST be fulfilled by the **PEP**. If the **PEP** does not understand or cannot  
2918 fulfill an **obligation**, then the action of the PEP is determined by its bias, see section 7.2. See  
2919 Section 7.18 for a description of how the set of **obligations** to be returned by the **PDP** is  
2920 determined.

2921 <AssociatedAdvice> [Optional]

2922 A list of **advice** that provide supplemental information to the **PEP**. If the **PEP** does not  
2923 understand an **advice**, the PEP may safely ignore the **advice**. See Section 7.18 for a description  
2924 of how the set of **advice** to be returned by the **PDP** is determined.

2925 <Attributes> [Optional]

2926 A list of **attributes** that were part of the request. The choice of which **attributes** are included here  
2927 is made with the IncludeInResult attribute of the <Attribute> elements of the request. See  
2928 section 5.46.

2929 <PolicyIdentifierList> [Optional]

2930 If the ReturnPolicyIdList attribute in the <Request> is true (see section 5.42), a **PDP** that  
2931 implements this optional feature MUST return a list which includes the identifiers of all **policies**  
2932 which were found to be fully applicable, whether or not the <Effect> was the same or different  
2933 from the <Decision>. The list MAY include the identifiers of other policies which are currently in



2934 force, as long as no policies required for the decision are omitted. A **PDP** MAY satisfy this  
2935 requirement by including all policies currently in force, or by including all policies which were  
2936 evaluated in making the decision, or by including all policies which did not evaluate to  
2937 “NotApplicable”, or by any other algorithm which does not omit any policies which contributed to  
2938 the decision. However, a decision which returns “NotApplicable” MUST return an empty list.  
2939

## 2940 5.49 Element <PolicyIdentifierList>

2941 The <PolicyIdentifierList> element contains a list of **policy** and **policy set** identifiers of **policies**  
2942 which have been applicable to a request. The list is unordered.

```
2943 <xs:element name="PolicyIdentifierList"  
2944   type="xacml:PolicyIdentifierListType"/>  
2945 <xs:complexType name="PolicyIdentifierListType">  
2946   <xs:choice minOccurs="0" maxOccurs="unbounded">  
2947     <xs:element ref="xacml:PolicyIdReference"/>  
2948     <xs:element ref="xacml:PolicySetIdReference"/>  
2949   </xs:choice>  
2950 </xs:complexType>
```

2951 The <PolicyIdentifierList> element is of PolicyIdentifierListType complex type.

2952 The <PolicyIdentifierList> element contains the following elements.

2953 <PolicyIdReference> [Any number]

2954 The identifier and version of a **policy** which was applicable to the request. See section 5.11. The  
2955 <PolicyIdReference> element MUST use the Version attribute to specify the version and  
2956 MUST NOT use the LatestVersion or EarliestVersion attributes.

2957 <PolicySetIdReference> [Any number]

2958 The identifier and version of a **policy set** which was applicable to the request. See section 5.10.  
2959 The <PolicySetIdReference> element MUST use the Version attribute to specify the  
2960 version and MUST NOT use the LatestVersion or EarliestVersion attributes.

## 2961 5.50 Element <MultiRequests>

2962 The <MultiRequests> element contains a list of requests by reference to <Attributes> elements in  
2963 the enclosing <Request> element. The semantics of this element are defined in [Multi]. Support for this  
2964 element is optional. If an implementation does not support this element, but receives it, the  
2965 implementation MUST generate an “Indeterminate” response.

```
2966 <xs:element name="MultiRequests" type="xacml:MultiRequestsType"/>  
2967 <xs:complexType name="MultiRequestsType">  
2968   <xs:sequence>  
2969     <xs:element ref="xacml:RequestReference" maxOccurs="unbounded"/>  
2970   </xs:sequence>  
2971 </xs:complexType>
```

2972 The <MultiRequests> element contains the following elements.

2973 <RequestReference> [one to many]

2974 Defines a request instance by reference to <Attributes> elements in the enclosing  
2975 <Request> element. See section 5.51.

## 2976 5.51 Element <RequestReference>

2977 The <RequestReference> element defines an instance of a request in terms of references to  
2978 <Attributes> elements. The semantics of this element are defined in [Multi]. Support for this element  
2979 is optional.



```

2980 <xs:element name="RequestReference" type="xacml:RequestReference" />
2981 <xs:complexType name="RequestReferenceType">
2982   <xs:sequence>
2983     <xs:element ref="xacml:AttributesReference" maxOccurs="unbounded" />
2984   </xs:sequence>
2985 </xs:complexType>

```

2986 The <RequestReference> element contains the following elements.

2987 <AttributesReference> [one to many]

2988       A reference to an <Attributes> element in the enclosing <Request> element. See section  
2989       5.52.

## 2990 5.52 Element <AttributesReference>

2991 The <AttributesReference> element makes a reference to an <Attributes> element. The  
2992 meaning of this element is defined in [Multi]. Support for this element is optional.

```

2993 <xs:element name="AttributesReference" type="xacml:AttributesReference" />
2994 <xs:complexType name="AttributesReferenceType">
2995   <xs:attribute name="ReferenceId" type="xs:IDREF" use="required" />
2996 </xs:complexType>

```

2997 The <AttributesReference> element contains the following attributes.

2998 ReferenceId [required]

2999       A reference to the xml:id attribute of an <Attributes> element in the enclosing <Request>  
3000       element.

## 3001 5.53 Element <Decision>

3002 The <Decision> element contains the result of *policy* evaluation.

```

3003 <xs:element name="Decision" type="xacml:DecisionType" />
3004 <xs:simpleType name="DecisionType">
3005   <xs:restriction base="xs:string">
3006     <xs:enumeration value="Permit" />
3007     <xs:enumeration value="Deny" />
3008     <xs:enumeration value="Indeterminate" />
3009     <xs:enumeration value="NotApplicable" />
3010   </xs:restriction>
3011 </xs:simpleType>

```

3012 The <Decision> element is of DecisionType simple type.

3013 The values of the <Decision> element have the following meanings:

3014       “Permit”: the requested **access** is permitted.

3015       “Deny”: the requested **access** is denied.

3016       “Indeterminate”: the **PDP** is unable to evaluate the requested **access**. Reasons for such inability  
3017       include: missing **attributes**, network errors while retrieving **policies**, division by zero during  
3018       **policy** evaluation, syntax errors in the **decision request** or in the **policy**, etc.

3019       “NotApplicable”: the **PDP** does not have any **policy** that applies to this **decision request**.

## 3020 5.54 Element <Status>

3021 The <Status> element represents the status of the **authorization decision** result.

```

3022 <xs:element name="Status" type="xacml:StatusType" />
3023 <xs:complexType name="StatusType">
3024   <xs:sequence>

```

3025  
3026  
3027  
3028  
3029

```
<xs:element ref="xacml:StatusCode"/>
<xs:element ref="xacml:StatusMessage" minOccurs="0"/>
<xs:element ref="xacml:StatusDetail" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
```

3030 The <Status> element is of StatusType complex type.

3031 The <Status> element contains the following elements:

3032 <StatusCode> [Required]

3033 Status code.

3034 <StatusMessage> [Optional]

3035 A status message describing the status code.

3036 <StatusDetail> [Optional]

3037 Additional status information.

## 3038 5.55 Element <StatusCode>

3039 The <StatusCode> element contains a major status code value and an optional recursive series of  
3040 minor status codes.

3041  
3042  
3043  
3044  
3045  
3046  
3047

```
<xs:element name="StatusCode" type="xacml:StatusCodeType"/>
<xs:complexType name="StatusCodeType">
  <xs:sequence>
    <xs:element ref="xacml:StatusCode" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="Value" type="xs:anyURI" use="required"/>
</xs:complexType>
```

3048 The <StatusCode> element is of StatusCodeType complex type.

3049 The <StatusCode> element contains the following attributes and elements:

3050 Value [Required]

3051 See Section B.8 for a list of values.

3052 <StatusCode> [Any Number]

3053 Minor status code. This status code qualifies its parent status code.

## 3054 5.56 Element <StatusMessage>

3055 The <StatusMessage> element is a free-form description of the status code.

3056

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

3057 The <StatusMessage> element is of xs:string type.

## 3058 5.57 Element <StatusDetail>

3059 The <StatusDetail> element qualifies the <Status> element with additional information.

3060  
3061  
3062  
3063  
3064  
3065  
3066

```
<xs:element name="StatusDetail" type="xacml:StatusDetailType"/>
<xs:complexType name="StatusDetailType">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

3067 The <StatusDetail> element is of StatusDetailType complex type.

3068 The <StatusDetail> element allows arbitrary XML content.

3069 Inclusion of a <StatusDetail> element is optional. However, if a **PDP** returns one of the following  
 3070 XACML-defined <StatusCode> values and includes a <StatusDetail> element, then the following  
 3071 rules apply.

3072 urn:oasis:names:tc:xacml:1.0:status:ok

3073 A **PDP** MUST NOT return a <StatusDetail> element in conjunction with the “ok” status value.

3074 urn:oasis:names:tc:xacml:1.0:status:missing-attribute

3075 A **PDP** MAY choose not to return any <StatusDetail> information or MAY choose to return a  
 3076 <StatusDetail> element containing one or more <MissingAttributeDetail> elements.

3077 urn:oasis:names:tc:xacml:1.0:status:syntax-error

3078 A **PDP** MUST NOT return a <StatusDetail> element in conjunction with the “syntax-error” status  
 3079 value. A syntax error may represent either a problem with the **policy** being used or with the request  
 3080 **context**. The **PDP** MAY return a <StatusMessage> describing the problem.

3081 urn:oasis:names:tc:xacml:1.0:status:processing-error

3082 A **PDP** MUST NOT return <StatusDetail> element in conjunction with the “processing-error” status  
 3083 value. This status code indicates an internal problem in the **PDP**. For security reasons, the **PDP** MAY  
 3084 choose to return no further information to the **PEP**. In the case of a divide-by-zero error or other  
 3085 computational error, the **PDP** MAY return a <StatusMessage> describing the nature of the error.

## 3086 5.58 Element <MissingAttributeDetail>

3087 The <MissingAttributeDetail> element conveys information about **attributes** required for **policy**  
 3088 evaluation that were missing from the request **context**.

```

3089 <xs:element name="MissingAttributeDetail"
3090 type="xacml:MissingAttributeDetailType"/>
3091 <xs:complexType name="MissingAttributeDetailType">
3092 <xs:sequence>
3093 <xs:element ref="xacml:AttributeValue" minOccurs="0"
3094 maxOccurs="unbounded"/>
3095 </xs:sequence>
3096 <xs:attribute name="Category" type="xs:anyURI" use="required"/>
3097 <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
3098 <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
3099 <xs:attribute name="Issuer" type="xs:string" use="optional"/>
3100 </xs:complexType>
  
```

3101 The <MissingAttributeDetail> element is of MissingAttributeDetailType complex type.

3102 The <MissingAttributeDetail> element contains the following attributes and elements:

3103 <AttributeValue> [Optional]

3104 The required value of the missing **attribute**.

3105 Category [Required]

3106 The category identifier of the missing **attribute**.

3107 AttributeId [Required]

3108 The identifier of the missing **attribute**.

3109 DataType [Required]

3110 The data-type of the missing **attribute**.

3111 Issuer [Optional]

3112 This attribute, if supplied, SHALL specify the required Issuer of the missing **attribute**.

3113 If the **PDP** includes <AttributeValue> elements in the <MissingAttributeDetail> element, then  
3114 this indicates the acceptable values for that **attribute**. If no <AttributeValue> elements are included,  
3115 then this indicates the names of **attributes** that the **PDP** failed to resolve during its evaluation. The list of  
3116 **attributes** may be partial or complete. There is no guarantee by the **PDP** that supplying the missing  
3117 values or **attributes** will be sufficient to satisfy the **policy**.

3118

## 6 XPath 2.0 definitions

3119 The XPath 2.0 specification leaves a number of aspects of behavior implementation defined. This section  
3120 defines how XPath 2.0 SHALL behave when hosted in XACML.

3121 <http://www.w3.org/TR/2007/REC-xpath20-20070123/#id-impl-defined-items> defines the following items:

- 3122 1. The version of Unicode that is used to construct expressions.  
3123 XACML leaves this implementation defined. It is RECOMMENDED that the latest version is used.
- 3124 2. The statically-known collations.  
3125 XACML leaves this implementation defined.
- 3126 3. The implicit timezone.  
3127 XACML defined the implicit time zone as UTC.
- 3128 4. The circumstances in which warnings are raised, and the ways in which warnings are handled.  
3129 XACML leaves this implementation defined.
- 3130 5. The method by which errors are reported to the external processing environment.  
3131 An XPath error causes an XACML Indeterminate value in the element where the XPath error  
3132 occurs. The StatusCode value SHALL be "urn:oasis:names:tc:xacml:1.0:status:processing-error".  
3133 Implementations MAY provide additional details about the error in the response or by some other  
3134 means.
- 3135 6. Whether the implementation is based on the rules of XML 1.0 or 1.1.  
3136 XACML is based on XML 1.0.
- 3137 7. Whether the implementation supports the namespace axis.  
3138 XACML leaves this implementation defined. It is RECOMMENDED that users of XACML do not  
3139 make use of the namespace axis.
- 3140 8. Any static typing extensions supported by the implementation, if the Static Typing Feature is  
3141 supported.  
3142 XACML leaves this implementation defined.

3143

3144 <http://www.w3.org/TR/2007/REC-xpath-datamodel-20070123/#implementation-defined> defines the  
3145 following items:

- 3146 1. Support for additional user-defined or implementation-defined types is implementation-defined.  
3147 It is RECOMMENDED that implementations of XACML do not define any additional types and it is  
3148 RECOMMENDED that users of XACML do not make user of any additional types.
- 3149 2. Some typed values in the data model are undefined. Attempting to access an undefined property  
3150 is always an error. Behavior in these cases is implementation-defined and the host language is  
3151 responsible for determining the result.  
3152 An XPath error causes an XACML Indeterminate value in the element where the XPath error  
3153 occurs. The StatusCode value SHALL be "urn:oasis:names:tc:xacml:1.0:status:processing-error".  
3154 Implementations MAY provide additional details about the error in the response or by some other  
3155 means.

3156

3157 <http://www.w3.org/TR/2007/REC-xpath-functions-20070123/#impl-def> defines the following items:

- 3158 1. The destination of the trace output is implementation-defined.  
3159 XACML leaves this implementation defined.
- 3160 2. For xs:integer operations, implementations that support limited-precision integer operations must  
3161 either raise an error [err:FOAR0002] or provide an implementation-defined mechanism that  
3162 allows users to choose between raising an error and returning a result that is modulo the largest  
3163 representable integer value.  
3164 XACML leaves this implementation defined. If an implementation chooses to raise an error, the

- 3165            StatusCode value SHALL be “urn:oasis:names:tc:xacml:1.0:status:processing-error”.
- 3166            Implementations MAY provide additional details about the error in the response or by some other
- 3167            means.
- 3168            3. For xs:decimal values the number of digits of precision returned by the numeric operators is
- 3169            implementation-defined.
- 3170            XACML leaves this implementation defined.
- 3171            4. If the number of digits in the result of a numeric operation exceeds the number of digits that the
- 3172            implementation supports, the result is truncated or rounded in an implementation-defined manner.
- 3173            XACML leaves this implementation defined.
- 3174            5. It is implementation-defined which version of Unicode is supported.
- 3175            XACML leaves this implementation defined. It is RECOMMENDED that the latest version is used.
- 3176            6. For fn:normalize-unicode, conforming implementations must support normalization form "NFC"
- 3177            and may support normalization forms "NFD", "NFKC", "NFKD", "FULLY-NORMALIZED". They
- 3178            may also support other normalization forms with implementation-defined semantics.
- 3179            XACML leaves this implementation defined.
- 3180            7. The ability to decompose strings into collation units suitable for substring matching is an
- 3181            implementation-defined property of a collation.
- 3182            XACML leaves this implementation defined.
- 3183            8. All minimally conforming processors must support year values with a minimum of 4 digits (i.e.,
- 3184            YYYY) and a minimum fractional second precision of 1 millisecond or three digits (i.e., s.sss).
- 3185            However, conforming processors may set larger implementation-defined limits on the maximum
- 3186            number of digits they support in these two situations.
- 3187            XACML leaves this implementation defined, and it is RECOMMENDED that users of XACML do
- 3188            not expect greater limits and precision.
- 3189            9. The result of casting a string to xs:decimal, when the resulting value is not too large or too small
- 3190            but nevertheless has too many decimal digits to be accurately represented, is implementation-
- 3191            defined.
- 3192            XACML leaves this implementation defined.
- 3193            10. Various aspects of the processing provided by fn:doc are implementation-defined.
- 3194            Implementations may provide external configuration options that allow any aspect of the
- 3195            processing to be controlled by the user.
- 3196            XACML leaves this implementation defined.
- 3197            11. The manner in which implementations provide options to weaken the stable characteristic of
- 3198            fn:collection and fn:doc are implementation-defined.
- 3199            XACML leaves this implementation defined.

---

## 3200 7 Functional requirements

3201 This section specifies certain functional requirements that are not directly associated with the production  
3202 or consumption of a particular XACML element.

3203 Note that in each case an implementation is conformant as long as it produces the same result as is  
3204 specified here, regardless of how and in what order the implementation behaves internally.

### 3205 7.1 Unicode issues

#### 3206 7.1.1 Normalization

3207 In Unicode, some equivalent characters can be represented by more than one different Unicode  
3208 character sequence. See [CMF]. The process of converting Unicode strings into equivalent character  
3209 sequences is called "normalization" [UAX15]. Some operations, such as string comparison, are sensitive  
3210 to normalization. An operation is normalization-sensitive if its output(s) are different depending on the  
3211 state of normalization of the input(s); if the output(s) are textual, they are deemed different only if they  
3212 would remain different were they to be normalized.

3213 For more information on normalization see [CM].

3214 An XACML implementation MUST behave as if each normalization-sensitive operation normalizes input  
3215 strings into Unicode Normalization Form C ("NFC"). An implementation MAY use some other form of  
3216 internal processing (such as using a non-Unicode, "legacy" character encoding) as long as the externally  
3217 visible results are identical to this specification.

#### 3218 7.1.2 Version of Unicode

3219 The version of Unicode used by XACML is implementation defined. It is RECOMMENDED that the latest  
3220 version is used. Also note security issues in section 9.3.

### 3221 7.2 Policy enforcement point

3222 This section describes the requirements for the *PEP*.

3223 An application functions in the role of the *PEP* if it guards *access* to a set of *resources* and asks the  
3224 *PDP* for an *authorization decision*. The *PEP* MUST abide by the *authorization decision* as described  
3225 in one of the following sub-sections

3226 In any case any *advice* in the *decision* may be safely ignored by the *PEP*.

#### 3227 7.2.1 Base PEP

3228 If the *decision* is "Permit", then the *PEP* SHALL permit *access*. If *obligations* accompany the *decision*,  
3229 then the *PEP* SHALL permit *access* only if it understands and it can and will discharge those  
3230 *obligations*.

3231 If the *decision* is "Deny", then the *PEP* SHALL deny *access*. If *obligations* accompany the *decision*,  
3232 then the *PEP* shall deny *access* only if it understands, and it can and will discharge those *obligations*.

3233 If the *decision* is "Not Applicable", then the *PEP*'s behavior is undefined.

3234 If the *decision* is "Indeterminate", then the *PEP*'s behavior is undefined.

#### 3235 7.2.2 Deny-biased PEP

3236 If the *decision* is "Permit", then the *PEP* SHALL permit *access*. If *obligations* accompany the *decision*,  
3237 then the *PEP* SHALL permit *access* only if it understands and it can and will discharge those  
3238 *obligations*.



3239 All other **decisions** SHALL result in the denial of **access**.

3240 Note: other actions, e.g. consultation of additional **PDPs**, reformulation/resubmission of  
3241 the **decision request**, etc., are not prohibited.

### 3242 7.2.3 Permit-biased PEP

3243 If the **decision** is "Deny", then the **PEP** SHALL deny **access**. If **obligations** accompany the **decision**,  
3244 then the **PEP** shall deny **access** only if it understands, and it can and will discharge those **obligations**.

3245 All other **decisions** SHALL result in the permission of **access**.

3246 Note: other actions, e.g. consultation of additional **PDPs**, reformulation/resubmission of  
3247 the **decision request**, etc., are not prohibited.

## 3248 7.3 Attribute evaluation

3249 **Attributes** are represented in the request **context** by the **context handler**, regardless of whether or not  
3250 they appeared in the original **decision request**, and are referred to in the **policy** by attribute designators  
3251 and attribute selectors. A **named attribute** is the term used for the criteria that the specific attribute  
3252 designators use to refer to particular **attributes** in the <Attributes> elements of the request **context**.

### 3253 7.3.1 Structured attributes

3254 <AttributeValue> elements MAY contain an instance of a structured XML data-type, for example  
3255 <ds:KeyInfo>. XACML 3.0 supports several ways for comparing the contents of such elements.

3256 1. In some cases, such elements MAY be compared using one of the XACML string functions, such  
3257 as "string-regexp-match", described below. This requires that the element be given the data-type  
3258 "http://www.w3.org/2001/XMLSchema#string". For example, a structured data-type that is  
3259 actually a ds:KeyInfo/KeyName would appear in the **Context** as:

```
3260 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">  
3261 <ds:KeyName>jhibbert-key</ds:KeyName>  
3262 </AttributeValue>
```

3263 In general, this method will not be adequate unless the structured data-type is quite simple.

3264 2. The structured **attribute** MAY be made available in the <Content> element of the appropriate  
3265 **attribute** category and an <AttributeSelector> element MAY be used to select the contents  
3266 of a leaf sub-element of the structured data-type by means of an XPath expression. That value  
3267 MAY then be compared using one of the supported XACML functions appropriate for its primitive  
3268 data-type. This method requires support by the **PDP** for the optional XPath expressions feature.

3269 3. The structured **attribute** MAY be made available in the <Content> element of the appropriate  
3270 **attribute** category and an <AttributeSelector> element MAY be used to select any node in  
3271 the structured data-type by means of an XPath expression. This node MAY then be compared  
3272 using one of the XPath-based functions described in Section A.3.15. This method requires  
3273 support by the **PDP** for the optional XPath expressions and XPath functions features.

3274 See also Section 7.3.

### 3275 7.3.2 Attribute bags

3276 XACML defines implicit collections of its data-types. XACML refers to a collection of values that are of a  
3277 single data-type as a **bag**. **Bags** of data-types are needed because selections of nodes from an XML  
3278 **resource** or XACML request **context** may return more than one value.

3279 The <AttributeSelector> element uses an XPath expression to specify the selection of data from  
3280 free form XML. The result of an XPath expression is termed a node-set, which contains all the nodes  
3281 from the XML content that match the **predicate** in the XPath expression. Based on the various indexing  
3282 functions provided in the XPath specification, it SHALL be implied that a resultant node-set is the

3283 collection of the matching nodes. XACML also defines the <AttributeDesignator> element to have  
3284 the same matching methodology for **attributes** in the XACML request **context**.

3285 The values in a **bag** are not ordered, and some of the values may be duplicates. There SHALL be no  
3286 notion of a **bag** containing **bags**, or a **bag** containing values of differing types; i.e., a **bag** in XACML  
3287 SHALL contain only values that are of the same data-type.

### 3288 7.3.3 Multivalued attributes

3289 If a single <Attribute> element in a request **context** contains multiple <AttributeValue> child  
3290 elements, then the **bag** of values resulting from evaluation of the <Attribute> element MUST be  
3291 identical to the **bag** of values that results from evaluating a **context** in which each <AttributeValue>  
3292 element appears in a separate <Attribute> element, each carrying identical meta-data.

### 3293 7.3.4 Attribute Matching

3294 A **named attribute** includes specific criteria with which to match **attributes** in the **context**. An **attribute**  
3295 specifies a **Category**, **AttributeId** and **DataType**, and a **named attribute** also specifies the  
3296 **Issuer**. A **named attribute** SHALL match an **attribute** if the values of their respective **Category**,  
3297 **AttributeId**, **DataType** and optional **Issuer** attributes match. The **Category** of the **named**  
3298 **attribute** MUST match, by **identifier equality**, the **Category** of the corresponding **context attribute**.  
3299 The **AttributeId** of the **named attribute** MUST match, by **identifier equality**, the **AttributeId** of  
3300 the corresponding **context attribute**. The **DataType** of the **named attribute** MUST match, by **identifier**  
3301 **equality**, the **DataType** of the corresponding **context attribute**. If **Issuer** is supplied in the **named**  
3302 **attribute**, then it MUST match, using the urn:oasis:names:tc:xacml:1.0:function:string-equal function, the  
3303 **Issuer** of the corresponding **context attribute**. If **Issuer** is not supplied in the **named attribute**, then  
3304 the matching of the **context attribute** to the **named attribute** SHALL be governed by **AttributeId** and  
3305 **DataType** alone, regardless of the presence, absence, or actual value of **Issuer** in the corresponding  
3306 **context attribute**. In the case of an attribute selector, the matching of the **attribute** to the **named**  
3307 **attribute** SHALL be governed by the XPath expression and **DataType**.

### 3308 7.3.5 Attribute Retrieval

3309 The **PDP** SHALL request the values of **attributes** in the request **context** from the **context handler**. The  
3310 **context handler** MAY also add **attributes** to the request **context** without the **PDP** requesting them. The  
3311 **PDP** SHALL reference the **attributes** as if they were in a physical request **context** document, but the  
3312 **context handler** is responsible for obtaining and supplying the requested values by whatever means it  
3313 deems appropriate, including by retrieving them from one or more Policy Information Points. The **context**  
3314 **handler** SHALL return the values of **attributes** that match the attribute designator or attribute selector  
3315 and form them into a **bag** of values with the specified data-type. If no **attributes** from the request  
3316 **context** match, then the **attribute** SHALL be considered missing. If the **attribute** is missing, then  
3317 **MustBePresent** governs whether the attribute designator or attribute selector returns an empty **bag** or  
3318 an “Indeterminate” result. If **MustBePresent** is “False” (default value), then a missing **attribute** SHALL  
3319 result in an empty **bag**. If **MustBePresent** is “True”, then a missing **attribute** SHALL result in  
3320 “Indeterminate”. This “Indeterminate” result SHALL be handled in accordance with the specification of the  
3321 encompassing expressions, **rules**, **policies** and **policy sets**. If the result is “Indeterminate”, then the  
3322 **AttributeId**, **DataType** and **Issuer** of the **attribute** MAY be listed in the **authorization decision** as  
3323 described in Section 7.17. However, a **PDP** MAY choose not to return such information for security  
3324 reasons.

3325 Regardless of any dynamic modifications of the request **context** during policy evaluation, the **PDP**  
3326 SHALL behave as if each bag of **attribute** values is fully populated in the **context** before it is first tested,  
3327 and is thereafter immutable during evaluation. (That is, every subsequent test of that **attribute** shall use  
3328 the same bag of values that was initially tested.)

3329 **7.3.6 Environment Attributes**

3330 Standard *environment attributes* are listed in Section B.7. If a value for one of these *attributes* is  
3331 supplied in the *decision request*, then the *context handler* SHALL use that value. Otherwise, the  
3332 *context handler* SHALL supply a value. In the case of date and time *attributes*, the supplied value  
3333 SHALL have the semantics of the "date and time that apply to the *decision request*".

3334 **7.3.7 AttributeSelector evaluation**

3335 An <AttributeSelector> element will be evaluated according to the following processing model.

3336

3337 NOTE: It is not necessary for an implementation to actually follow these steps. It is only  
3338 necessary to produce results identical to those that would be produced by following these  
3339 steps.

- 3340 1. If the *attributes* category given by the *Category* attribute is not found or does not have a  
3341 <Content> child element, then the return value is either "Indeterminate" or an empty *bag* as  
3342 determined by the *MustBePresent* attribute; otherwise, construct an XML data structure  
3343 suitable for xpath processing from the <Content> element in the attributes category given by the  
3344 *Category* attribute. The data structure shall be constructed so that the document node of this  
3345 structure contains a single document element which corresponds to the single child element of  
3346 the <Content> element. The constructed data structure shall be equivalent to one that would  
3347 result from parsing a stand-alone XML document consisting of the contents of the <Content>  
3348 element (including any comment and processing-instruction markup). Namespace declarations  
3349 which are not "visibly utilized", as defined by [exc-c14n], MAY not be present and MUST NOT be  
3350 utilized by the XPath expression in step 3. The data structure must meet the requirements of the  
3351 applicable xpath version.
- 3352 2. Select a context node for xpath processing from this data structure. If there is a  
3353 *ContextSelectorId* attribute, the context node shall be the node selected by applying the  
3354 XPath expression given in the *attribute* value of the designated *attribute* (in the *attributes*  
3355 category given by the <AttributeSelector> *Category* attribute). It shall be an error if this  
3356 evaluation returns no node or more than one node, in which case the return value MUST be an  
3357 "Indeterminate" with a status code "urn:oasis:names:tc:xacml:1.0:status:syntax-error". If there is  
3358 no *ContextSelectorId*, the document node of the data structure shall be the context node.
- 3359 3. Evaluate the XPath expression given in the *Path* attribute against the xml data structure, using  
3360 the context node selected in the previous step. It shall be an error if this evaluation returns  
3361 anything other than a sequence of nodes (possibly empty), in which case the  
3362 <AttributeSelector> MUST return "Indeterminate" with a status code  
3363 "urn:oasis:names:tc:xacml:1.0:status:syntax-error". If the evaluation returns an empty sequence  
3364 of nodes, then the return value is either "Indeterminate" or an empty *bag* as determined by the  
3365 *MustBePresent* attribute.
- 3366 4. If the data type is a primitive data type, convert the text value of each selected node to the  
3367 desired data type, as specified in the *DataType* attribute. Each value shall be constructed using  
3368 the appropriate constructor function from [XF] Section 5 listed below, corresponding to the  
3369 specified data type.

- 3370
- 3371 xs:string()
- 3372 xs:boolean()
- 3373 xs:integer()
- 3374 xs:double()
- 3375 xs:dateTime()
- 3376 xs:date()
- 3377 xs:time()
- 3378 xs:hexBinary()
- 3379 xs:base64Binary()

3380 xs:anyURI()  
3381 xs:yearMonthDuration()  
3382 xs:dayTimeDuration()  
3383

3384 If the `DataType` is not one of the primitive types listed above, then the return values shall be  
3385 constructed from the nodeset in a manner specified by the particular `DataType` extension  
3386 specification. If the data type extension does not specify an appropriate constructor function, then  
3387 the `<AttributeSelector>` MUST return "Indeterminate" with a status code  
3388 "urn:oasis:names:tc:xacml:1.0:status:syntax-error".  
3389

3390 If an error occurs when converting the values returned by the XPath expression to the specified  
3391 `DataType`, then the result of the `<AttributeSelector>` MUST be "Indeterminate", with a  
3392 status code "urn:oasis:names:tc:xacml:1.0:status:processing-error"

## 3393 7.4 Expression evaluation

3394 XACML specifies expressions in terms of the elements listed below, of which the `<Apply>` and  
3395 `<Condition>` elements recursively compose greater expressions. Valid expressions SHALL be type  
3396 correct, which means that the types of each of the elements contained within `<Apply>` elements SHALL  
3397 agree with the respective argument types of the function that is named by the `FunctionId` attribute.  
3398 The resultant type of the `<Apply>` element SHALL be the resultant type of the function, which MAY be  
3399 narrowed to a primitive data-type, or a **bag** of a primitive data-type, by type-unification. XACML defines  
3400 an evaluation result of "Indeterminate", which is said to be the result of an invalid expression, or an  
3401 operational error occurring during the evaluation of the expression.

3402 XACML defines these elements to be in the substitution group of the `<Expression>` element:

- 3403 • `<xacml:AttributeValue>`
- 3404 • `<xacml:AttributeDesignator>`
- 3405 • `<xacml:AttributeSelector>`
- 3406 • `<xacml:Apply>`
- 3407 • `<xacml:Function>`
- 3408 • `<xacml:VariableReference>`

## 3409 7.5 Arithmetic evaluation

3410 IEEE 754 [IEEE754] specifies how to evaluate arithmetic functions in a context, which specifies defaults  
3411 for precision, rounding, etc. XACML SHALL use this specification for the evaluation of all integer and  
3412 double functions relying on the Extended Default Context, enhanced with double precision:

3413 flags - all set to 0

3414 trap-enablers - all set to 0 (IEEE 854 §7) with the exception of the "division-by-zero" trap enabler,  
3415 which SHALL be set to 1

3416 precision - is set to the designated double precision

3417 rounding - is set to round-half-even (IEEE 854 §4.1)

## 3418 7.6 Match evaluation

3419 The **attribute** matching element `<Match>` appears in the `<Target>` element of **rules**, **policies** and  
3420 **policy sets**.

3421 This element represents a Boolean expression over **attributes** of the request **context**. A matching  
3422 element contains a `MatchId` attribute that specifies the function to be used in performing the match  
3423 evaluation, an `<AttributeValue>` and an `<AttributeDesignator>` or `<AttributeSelector>`  
3424 element that specifies the **attribute** in the **context** that is to be matched against the specified value.

3425 The `MatchId` attribute SHALL specify a function that takes two arguments, returning a result type of  
3426 "http://www.w3.org/2001/XMLSchema#boolean". The **attribute** value specified in the matching element  
3427 SHALL be supplied to the `MatchId` function as its first argument. An element of the **bag** returned by the  
3428 `<AttributeDesignator>` or `<AttributeSelector>` element SHALL be supplied to the `MatchId`  
3429 function as its second argument, as explained below. The `DataType` of the `<AttributeValue>`  
3430 SHALL match the data-type of the first argument expected by the `MatchId` function. The `DataType` of  
3431 the `<AttributeDesignator>` or `<AttributeSelector>` element SHALL match the data-type of the  
3432 second argument expected by the `MatchId` function.

3433 In addition, functions that are strictly within an extension to XACML MAY appear as a value for the  
3434 `MatchId` attribute, and those functions MAY use data-types that are also extensions, so long as the  
3435 extension function returns a Boolean result and takes two single base types as its inputs. The function  
3436 used as the value for the `MatchId` attribute SHOULD be easily indexable. Use of non-indexable or  
3437 complex functions may prevent efficient evaluation of **decision requests**.

3438 The evaluation semantics for a matching element is as follows. If an operational error were to occur while  
3439 evaluating the `<AttributeDesignator>` or `<AttributeSelector>` element, then the result of the  
3440 entire expression SHALL be "Indeterminate". If the `<AttributeDesignator>` or  
3441 `<AttributeSelector>` element were to evaluate to an empty **bag**, then the result of the expression  
3442 SHALL be "False". Otherwise, the `MatchId` function SHALL be applied between the  
3443 `<AttributeValue>` and each element of the **bag** returned from the `<AttributeDesignator>` or  
3444 `<AttributeSelector>` element. If at least one of those function applications were to evaluate to  
3445 "True", then the result of the entire expression SHALL be "True". Otherwise, if at least one of the function  
3446 applications results in "Indeterminate", then the result SHALL be "Indeterminate". Finally, if all function  
3447 applications evaluate to "False", then the result of the entire expression SHALL be "False".

3448 It is also possible to express the semantics of a **target** matching element in a **condition**. For instance,  
3449 the **target** match expression that compares a "**subject-name**" starting with the name "John" can be  
3450 expressed as follows:

```
3451 <Match  
3452 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">  
3453   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">  
3454     John.*  
3455   </AttributeValue>  
3456   <AttributeDesignator  
3457     Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-  
3458 subject"  
3459     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"  
3460     DataType="http://www.w3.org/2001/XMLSchema#string"/>  
3461 </Match>
```

3462 Alternatively, the same match semantics can be expressed as an `<Apply>` element in a **condition** by  
3463 using the "urn:oasis:names:tc:xacml:3.0:function:any-of" function, as follows:

```
3464 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">  
3465   <Function  
3466   FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match"/>  
3467   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">  
3468     John.*  
3469   </AttributeValue>  
3470   <AttributeDesignator  
3471     Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-  
3472 subject"  
3473     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"  
3474     DataType="http://www.w3.org/2001/XMLSchema#string"/>  
3475 </Apply>
```

3476 **7.7 Target evaluation**

3477 An empty **target** matches any request. Otherwise the **target** value SHALL be "Match" if all the AnyOf  
 3478 specified in the **target** match values in the request **context**. Otherwise, if any one of the AnyOf specified  
 3479 in the **target** is "No Match", then the **target** SHALL be "No Match". Otherwise, the **target** SHALL be  
 3480 "Indeterminate". The **target** match table is shown in Table 1.

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

3481 *Table 1 Target match table*

3482 The AnyOf SHALL match values in the request **context** if at least one of their <AllOf> elements  
 3483 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"

3484 *Table 2 AnyOf match table*

3485 An AllOf SHALL match a value in the request **context** if the value of all its <Match> elements is "True".

3486 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"

3487 *Table 3 AllOf match table*

3488 **7.8 VariableReference Evaluation**

3489 The <VariableReference> element references a single <VariableDefinition> element contained  
 3490 within the same <Policy> element. A <VariableReference> that does not reference a particular  
 3491 <VariableDefinition> element within the encompassing <Policy> element is called an undefined  
 3492 reference. **Policies** with undefined references are invalid.

3493 In any place where a <VariableReference> occurs, it has the effect as if the text of the  
 3494 <Expression> element defined in the <VariableDefinition> element replaces the  
 3495 <VariableReference> element. Any evaluation scheme that preserves this semantic is acceptable.  
 3496 For instance, the expression in the <VariableDefinition> element may be evaluated to a particular  
 3497 value and cached for multiple references without consequence. (I.e. the value of an <Expression>  
 3498 element remains the same for the entire **policy** evaluation.) This characteristic is one of the benefits of  
 3499 XACML being a declarative language.

3500 A variable reference containing circular references is invalid. The PDP MUST detect circular references  
 3501 either at policy loading time or during runtime evaluation. If the PDP detects a circular reference during



3502 runtime the variable reference evaluates to "Indeterminate" with status code  
 3503 urn:oasis:names:tc:xacml:1.0:status:processing-error.

## 3504 7.9 Condition evaluation

3505 The **condition** value SHALL be "True" if the <Condition> element is absent, or if it evaluates to "True".  
 3506 Its value SHALL be "False" if the <Condition> element evaluates to "False". The **condition** value  
 3507 SHALL be "Indeterminate", if the expression contained in the <Condition> element evaluates to  
 3508 "Indeterminate."

## 3509 7.10 Extended Indeterminate

3510 Some **combining algorithms** are defined in terms of an extended set of "Indeterminate" values. The  
 3511 extended set associated with the "Indeterminate" contains the potential effect values which could have  
 3512 occurred if there would not have been an error causing the "Indeterminate". The possible extended set  
 3513 "Indeterminate" values are

- 3514 • "Indeterminate{D}": an "Indeterminate" from a **policy** or **rule** which could have evaluated to "Deny",  
 3515 but not "Permit"
- 3516 • "Indeterminate{P}": an "Indeterminate" from a **policy** or **rule** which could have evaluated to "Permit",  
 3517 but not "Deny"
- 3518 • "Indeterminate{DP}": an "Indeterminate" from a **policy** or **rule** which could have evaluated to "Deny"  
 3519 or "Permit".

3520 The **combining algorithms** which are defined in terms of the extended "Indeterminate" make use of the  
 3521 additional information to allow for better treatment of errors in the algorithms.

3522 The final decision returned by a **PDP** cannot be an extended Indeterminate. Any such decision at the top  
 3523 level **policy** or **policy set** is returned as a plain Indeterminate in the response from the **PDP**.

3524 The tables in the following four sections define how extended "Indeterminate" values are produced during  
 3525 **Rule**, **Policy** and **PolicySet** evaluation.

## 3526 7.11 Rule evaluation

3527 A **rule** has a value that can be calculated by evaluating its contents. **Rule** evaluation involves separate  
 3528 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

3529 Table 4 Rule truth table.

## 3530 7.12 Policy evaluation

3531 The value of a **policy** SHALL be determined only by its contents, considered in relation to the contents of  
 3532 the request **context**. A **policy's** value SHALL be determined by evaluation of the **policy's target** and,  
 3533 according to the specified **rule-combining algorithm, rules**,.

3534 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

3535 *Table 5 Policy truth table*

3536 Note that none of the **rule-combining algorithms** defined by XACML 3.0 take parameters. However,  
 3537 non-standard combining algorithms MAY take parameters. In such a case, the values of these  
 3538 parameters associated with the **rules**, MUST be taken into account when evaluating the **policy**. The  
 3539 parameters and their types should be defined in the specification of the combining algorithm. If the  
 3540 implementation supports combiner parameters and if combiner parameters are present in a **policy**, then  
 3541 the parameter values MUST be supplied to the combining algorithm implementation.

### 3542 7.13 Policy Set evaluation

3543 The value of a **policy set** SHALL be determined by its contents, considered in relation to the contents of  
 3544 the request **context**. A **policy set**'s value SHALL be determined by evaluation of the **policy set**'s **target**,  
 3545 and, according to the specified **policy-combining algorithm**, **policies** and **policy sets**,

3546 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

3547 *Table 6 Policy set truth table*

3548 Note that none of the **policy-combining algorithms** defined by XACML 3.0 take parameters. However,  
 3549 non-standard combining algorithms MAY take parameters. In such a case, the values of these  
 3550 parameters associated with the **policies**, MUST be taken into account when evaluating the **policy set**.  
 3551 The parameters and their types should be defined in the specification of the combining algorithm. If the  
 3552 implementation supports combiner parameters and if combiner parameters are present in a **policy**, then  
 3553 the parameter values MUST be supplied to the combining algorithm implementation.

### 3554 7.14 Policy and Policy set value for Indeterminate Target

3555 If the **target** of a **policy** or **policy set** evaluates to “Indeterminate”, the value of the **policy** or **policy set**  
 3556 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}"
--------------------

3557 Table 7 The value of a **policy** or **policy set** when the target is "Indeterminate".

## 3558 7.15 PolicySetIdReference and PolicyIdReference evaluation

3559 A policy set id reference or a policy id reference is evaluated by resolving the reference and evaluating  
3560 the referenced policy set or policy.

3561 If resolving the reference fails, the reference evaluates to "Indeterminate" with status code  
3562 urn:oasis:names:tc:xacml:1.0:status:processing-error.

3563 A policy set id reference or a policy id reference containing circular references is invalid. The PDP MUST  
3564 detect circular references either at policy loading time or during runtime evaluation. If the PDP detects a  
3565 circular reference during runtime the reference evaluates to "Indeterminate" with status code  
3566 urn:oasis:names:tc:xacml:1.0:status:processing-error.

## 3567 7.16 Hierarchical resources

3568 It is often the case that a **resource** is organized as a hierarchy (e.g. file system, XML document). XACML  
3569 provides several optional mechanisms for supporting hierarchical **resources**. These are described in the  
3570 XACML Profile for Hierarchical Resources [**Hier**] and in the XACML Profile for Requests for Multiple  
3571 Resources [**Multi**].

## 3572 7.17 Authorization decision

3573 In relation to a particular **decision request**, the **PDP** is defined by a **policy-combining algorithm** and a  
3574 set of **policies** and/or **policy sets**. The **PDP** SHALL return a response **context** as if it had evaluated a  
3575 single **policy set** consisting of this **policy-combining algorithm** and the set of **policies** and/or **policy**  
3576 **sets**.

3577 The **PDP** MUST evaluate the **policy set** as specified in Sections 5 and 7. The **PDP** MUST return a  
3578 response **context**, with one <Decision> element of value "Permit", "Deny", "Indeterminate" or  
3579 "NotApplicable".

3580 If the **PDP** cannot make a **decision**, then an "Indeterminate" <Decision> element SHALL be returned.

## 3581 7.18 Obligations and advice

3582 A **rule**, **policy**, or **policy set** may contain one or more **obligation** or **advice** expressions. When such a  
3583 **rule**, **policy**, or **policy set** is evaluated, the **obligation** or **advice** expression SHALL be evaluated to an  
3584 **obligation** or **advice** respectively, which SHALL be passed up to the next level of evaluation (the  
3585 enclosing or referencing **policy**, **policy set**, or **authorization decision**) only if the result of the **rule**,  
3586 **policy**, or **policy set** being evaluated matches the value of the FulfillOn attribute of the **obligation** or  
3587 the AppliesTo attribute of the **advice**. If any of the **attribute** assignment expressions in an **obligation**  
3588 or **advice** expression with a matching FulfillOn or AppliesTo attribute evaluates to "Indeterminate",  
3589 then the whole **rule**, **policy**, or **policy set** SHALL be "Indeterminate". If the FulfillOn or AppliesTo  
3590 attribute does not match the result of the combining algorithm or the **rule** evaluation, then any  
3591 indeterminate in an **obligation** or **advice** expression has no effect.

3592 As a consequence of this procedure, no **obligations** or **advice** SHALL be returned to the **PEP** if the **rule**,  
3593 **policies**, or **policy sets** from which they are drawn are not evaluated, or if their evaluated result is  
3594 "Indeterminate" or "NotApplicable", or if the **decision** resulting from evaluating the **rule**, **policy**, or **policy**  
3595 **set** does not match the **decision** resulting from evaluating an enclosing **policy set**.

3596 If the **PDP**'s evaluation is viewed as a tree of **rules**, **policy sets** and **policies**, each of which returns  
3597 "Permit" or "Deny", then the set of **obligations** and **advice** returned by the **PDP** to the **PEP** will include  
3598 only the **obligations** and **advice** associated with those paths where the result at each level of evaluation  
3599 is the same as the result being returned by the **PDP**. In situations where any lack of determinism is  
3600 unacceptable, a deterministic combining algorithm, such as ordered-deny-overrides, should be used.

3601 Also see Section 7.2.

## 3602 7.19 Exception handling

3603 XACML specifies behavior for the **PDP** in the following situations.

### 3604 7.19.1 Unsupported functionality

3605 If the **PDP** attempts to evaluate a **policy set** or **policy** that contains an optional element type or function  
3606 that the **PDP** does not support, then the **PDP** SHALL return a <Decision> value of "Indeterminate". If a  
3607 <StatusCode> element is also returned, then its value SHALL be  
3608 "urn:oasis:names:tc:xacml:1.0:status:syntax-error" in the case of an unsupported element type, and  
3609 "urn:oasis:names:tc:xacml:1.0:status:processing-error" in the case of an unsupported function.

### 3610 7.19.2 Syntax and type errors

3611 If a **policy** that contains invalid syntax is evaluated by the XACML **PDP** at the time a **decision request** is  
3612 received, then the result of that **policy** SHALL be "Indeterminate" with a `StatusCode` value of  
3613 "urn:oasis:names:tc:xacml:1.0:status:syntax-error".

3614 If a **policy** that contains invalid static data-types is evaluated by the XACML **PDP** at the time a **decision**  
3615 **request** is received, then the result of that **policy** SHALL be "Indeterminate" with a `StatusCode` value of  
3616 "urn:oasis:names:tc:xacml:1.0:status:processing-error".

### 3617 7.19.3 Missing attributes

3618 The absence of matching **attributes** in the request **context** for any of the attribute designators attribute or  
3619 selectors that are found in the **policy** will result in an enclosing <AllOf> element to return a value of  
3620 "Indeterminate", if the designator or selector has the `MustBePresent` XML attribute set to true, as  
3621 described in Sections 5.29 and 5.30 and may result in a <Decision> element containing the  
3622 "Indeterminate" value. If, in this case a status code is supplied, then the value

3623 "urn:oasis:names:tc:xacml:1.0:status:missing-attribute"

3624 SHALL be used, to indicate that more information is needed in order for a definitive **decision** to be  
3625 rendered. In this case, the <Status> element MAY list the names and data-types of any **attributes** that  
3626 are needed by the **PDP** to refine its **decision** (see Section 5.58). A **PEP** MAY resubmit a refined request  
3627 **context** in response to a <Decision> element contents of "Indeterminate" with a status code of

3628 "urn:oasis:names:tc:xacml:1.0:status:missing-attribute"

3629 by adding **attribute** values for the **attribute** names that were listed in the previous response. When the  
3630 **PDP** returns a <Decision> element contents of "Indeterminate", with a status code of

3631 "urn:oasis:names:tc:xacml:1.0:status:missing-attribute",

3632 it MUST NOT list the names and data-types of any **attribute** for which values were supplied in the original  
3633 request. Note, this requirement forces the **PDP** to eventually return an **authorization decision** of  
3634 "Permit", "Deny", or "Indeterminate" with some other status code, in response to successively-refined  
3635 requests.

## 3636 7.20 Identifier equality

3637 XACML makes use of URIs and strings as identifiers. When such identifiers are compared for equality,  
3638 the comparison MUST be done so that the identifiers are equal if they have the same length and the  
3639 characters in the two identifiers are equal codepoint by codepoint.

3640 The following is a list of the identifiers which MUST use this definition of equality.

3641 The content of the element <XPathVersion>.

3642 The XML attribute `Value` in the element <StatusCode>.

- 3643 The XML attributes `Category`, `AttributeId`, `DataType` and `Issuer` in the element  
3644 `<MissingAttributeDetail>`.
- 3645 The XML attribute `Category` in the element `<Attributes>`.
- 3646 The XML attributes `AttributeId` and `Issuer` in the element `<Attribute>`.
- 3647 The XML attribute `ObligationId` in the element `<Obligation>`.
- 3648 The XML attribute `AdviceId` in the element `<Advice>`.
- 3649 The XML attributes `AttributeId` and `Category` in the element `<AttributeAssignment>`.
- 3650 The XML attribute `ObligationId` in the element `<ObligationExpression>`.
- 3651 The XML attribute `AdviceId` in the element `<AdviceExpression>`.
- 3652 The XML attributes `AttributeId`, `Category` and `Issuer` in the element  
3653 `<AttributeAssignmentExpression>`.
- 3654 The XML attributes `PolicySetId` and `PolicyCombiningAlgId` in the element `<PolicySet>`.
- 3655 The XML attribute `ParameterName` in the element `<CombinerParameter>`.
- 3656 The XML attribute `RuleIdRef` in the element `<RuleCombinerParameters>`.
- 3657 The XML attribute `PolicyIdRef` in the element `<PolicyCombinerParameters>`.
- 3658 The XML attribute `PolicySetIdRef` in the element `<PolicySetCombinerParameters>`.
- 3659 The anyURI in the content of the complex type `IdReferenceType`.
- 3660 The XML attributes `PolicyId` and `RuleCombiningAlgId` in the element `<Policy>`.
- 3661 The XML attribute `RuleId` in the element `<Rule>`.
- 3662 The XML attribute `MatchId` in the element `<Match>`.
- 3663 The XML attribute `VariableId` in the element `<VariableDefinition>`.
- 3664 The XML attribute `VariableId` in the element `<VariableReference>`.
- 3665 The XML attributes `Category`, `ContextSelectorId` and `DataType` in the element  
3666 `<AttributeSelector>`.
- 3667 The XML attributes `Category`, `AttributeId`, `DataType` and `Issuer` in the element  
3668 `<AttributeDesignator>`.
- 3669 The XML attribute `DataType` in the element `<AttributeValue>`.
- 3670 The XML attribute `FunctionId` in the element `<Function>`.
- 3671 The XML attribute `FunctionId` in the element `<Apply>`.
- 3672
- 3673 It is RECOMMENDED that extensions to XACML use the same definition of identifier equality for similar  
3674 identifiers.
- 3675 It is RECOMMENDED that extensions which define identifiers do not define identifiers which could be  
3676 easily misinterpreted by people as being subject to other kind of processing, such as URL character  
3677 escaping, before matching.

---

## 3678 8 XACML extensibility points (non-normative)

3679 This section describes the points within the XACML model and schema where extensions can be added.

### 3680 8.1 Extensible XML attribute types

3681 The following XML attributes have values that are URIs. These may be extended by the creation of new  
3682 URIs associated with new semantics for these attributes.

3683 `Category,`

3684 `AttributeId,`

3685 `DataType,`

3686 `FunctionId,`

3687 `MatchId,`

3688 `ObligationId,`

3689 `AdviceId,`

3690 `PolicyCombiningAlgId,`

3691 `RuleCombiningAlgId,`

3692 `StatusCode.`

3693 See Section 5 for definitions of these *attribute* types.

### 3694 8.2 Structured attributes

3695 `<AttributeValue>` elements MAY contain an instance of a structured XML data-type. Section 7.3.1  
3696 describes a number of standard techniques to identify data items within such a structured *attribute*.  
3697 Listed here are some additional techniques that require XACML extensions.

- 3698 1. For a given structured data-type, a community of XACML users MAY define new *attribute*  
3699 identifiers for each leaf sub-element of the structured data-type that has a type conformant with  
3700 one of the XACML-defined primitive data-types. Using these new *attribute* identifiers, the *PEPs*  
3701 or *context handlers* used by that community of users can flatten instances of the structured  
3702 data-type into a sequence of individual `<Attribute>` elements. Each such `<Attribute>`  
3703 element can be compared using the XACML-defined functions. Using this method, the structured  
3704 data-type itself never appears in an `<AttributeValue>` element.
- 3705 2. A community of XACML users MAY define a new function that can be used to compare a value of  
3706 the structured data-type against some other value. This method may only be used by *PDPs* that  
3707 support the new function.

---

## 3708 9 Security and privacy considerations (non- 3709 normative)

3710 This section identifies possible security and privacy compromise scenarios that should be considered  
3711 when implementing an XACML-based system. The section is informative only. It is left to the  
3712 implementer to decide whether these compromise scenarios are practical in their environment and to  
3713 select appropriate safeguards.

### 3714 9.1 Threat model

3715 We assume here that the adversary has access to the communication channel between the XACML  
3716 actors and is able to interpret, insert, delete, and modify messages or parts of messages.

3717 Additionally, an actor may use information from a former message maliciously in subsequent transactions.  
3718 It is further assumed that *rules* and *policies* are only as reliable as the actors that create and use them.  
3719 Thus it is incumbent on each actor to establish appropriate trust in the other actors upon which it relies.  
3720 Mechanisms for trust establishment are outside the scope of this specification.

3721 The messages that are transmitted between the actors in the XACML model are susceptible to attack by  
3722 malicious third parties. Other points of vulnerability include the *PEP*, the *PDP*, and the *PAP*. While some  
3723 of these entities are not strictly within the scope of this specification, their compromise could lead to the  
3724 compromise of *access control* enforced by the *PEP*.

3725 It should be noted that there are other components of a distributed system that may be compromised,  
3726 such as an operating system and the domain-name system (DNS) that are outside the scope of this  
3727 discussion of threat models. Compromise in these components may also lead to a policy violation.

3728 The following sections detail specific compromise scenarios that may be relevant to an XACML system.

#### 3729 9.1.1 Unauthorized disclosure

3730 XACML does not specify any inherent mechanisms to protect the confidentiality of the messages  
3731 exchanged between actors. Therefore, an adversary could observe the messages in transit. Under  
3732 certain security *policies*, disclosure of this information is a violation. Disclosure of *attributes* or the types  
3733 of *decision requests* that a *subject* submits may be a breach of privacy policy. In the commercial  
3734 sector, the consequences of unauthorized disclosure of personal data may range from embarrassment to  
3735 the custodian, to imprisonment and/or large fines in the case of medical or financial data.

3736 Unauthorized disclosure is addressed by confidentiality safeguards.

#### 3737 9.1.2 Message replay

3738 A message replay attack is one in which the adversary records and replays legitimate messages between  
3739 XACML actors. This attack may lead to denial of service, the use of out-of-date information or  
3740 impersonation.

3741 Prevention of replay attacks requires the use of message freshness safeguards.

3742 Note that encryption of the message does not mitigate a replay attack since the message is simply  
3743 replayed and does not have to be understood by the adversary.

#### 3744 9.1.3 Message insertion

3745 A message insertion attack is one in which the adversary inserts messages in the sequence of messages  
3746 between XACML actors.

3747 The solution to a message insertion attack is to use mutual authentication and message sequence  
3748 integrity safeguards between the actors. It should be noted that just using SSL mutual authentication is  
3749 not sufficient. This only proves that the other party is the one identified by the *subject* of the X.509



3750 certificate. In order to be effective, it is necessary to confirm that the certificate **subject** is authorized to  
3751 send the message.

#### 3752 **9.1.4 Message deletion**

3753 A message deletion attack is one in which the adversary deletes messages in the sequence of messages  
3754 between XACML actors. Message deletion may lead to denial of service. However, a properly designed  
3755 XACML system should not render an incorrect **authorization decision** as a result of a message deletion  
3756 attack.

3757 The solution to a message deletion attack is to use message sequence integrity safeguards between the  
3758 actors.

#### 3759 **9.1.5 Message modification**

3760 If an adversary can intercept a message and change its contents, then they may be able to alter an  
3761 **authorization decision**. A message integrity safeguard can prevent a successful message modification  
3762 attack.

#### 3763 **9.1.6 NotApplicable results**

3764 A result of "NotApplicable" means that the **PDP** could not locate a **policy** whose **target** matched the  
3765 information in the **decision request**. In general, it is highly recommended that a "Deny" **effect policy** be  
3766 used, so that when a **PDP** would have returned "NotApplicable", a result of "Deny" is returned instead.

3767 In some security models, however, such as those found in many web servers, an **authorization decision**  
3768 of "NotApplicable" is treated as equivalent to "Permit". There are particular security considerations that  
3769 must be taken into account for this to be safe. These are explained in the following paragraphs.

3770 If "NotApplicable" is to be treated as "Permit", it is vital that the matching algorithms used by the **policy** to  
3771 match elements in the **decision request** be closely aligned with the data syntax used by the applications  
3772 that will be submitting the **decision request**. A failure to match will result in "NotApplicable" and be  
3773 treated as "Permit". So an unintended failure to match may allow unintended **access**.

3774 Commercial http responders allow a variety of syntaxes to be treated equivalently. The "%" can be used  
3775 to represent characters by hex value. The URL path "/./" provides multiple ways of specifying the same  
3776 value. Multiple character sets may be permitted and, in some cases, the same printed character can be  
3777 represented by different binary values. Unless the matching algorithm used by the **policy** is sophisticated  
3778 enough to catch these variations, unintended **access** may be permitted.

3779 It may be safe to treat "NotApplicable" as "Permit" only in a closed environment where all applications that  
3780 formulate a **decision request** can be guaranteed to use the exact syntax expected by the **policies**. In a  
3781 more open environment, where **decision requests** may be received from applications that use any legal  
3782 syntax, it is strongly recommended that "NotApplicable" NOT be treated as "Permit" unless matching  
3783 **rules** have been very carefully designed to match all possible applicable inputs, regardless of syntax or  
3784 type variations. Note, however, that according to Section 7.2, a **PEP** must deny **access** unless it  
3785 receives an explicit "Permit" **authorization decision**.

#### 3786 **9.1.7 Negative rules**

3787 A negative **rule** is one that is based on a **predicate** not being "True". If not used with care, negative  
3788 **rules** can lead to policy violations, therefore some authorities recommend that they not be used.  
3789 However, negative **rules** can be extremely efficient in certain cases, so XACML has chosen to include  
3790 them. Nevertheless, it is recommended that they be used with care and avoided if possible.

3791 A common use for negative **rules** is to deny **access** to an individual or subgroup when their membership  
3792 in a larger group would otherwise permit them **access**. For example, we might want to write a **rule** that  
3793 allows all vice presidents to see the unpublished financial data, except for Joe, who is only a ceremonial  
3794 vice president and can be indiscreet in his communications. If we have complete control over the  
3795 administration of **subject attributes**, a superior approach would be to define "Vice President" and  
3796 "Ceremonial Vice President" as distinct groups and then define **rules** accordingly. However, in some



3797 environments this approach may not be feasible. (It is worth noting in passing that referring to individuals  
3798 in **rules** does not scale well. Generally, shared **attributes** are preferred.)

3799 If not used with care, negative **rules** can lead to policy violations in two common cases: when **attributes**  
3800 are suppressed and when the base group changes. An example of suppressed **attributes** would be if we  
3801 have a **policy** that **access** should be permitted, unless the **subject** is a credit risk. If it is possible that  
3802 the **attribute** of being a credit risk may be unknown to the **PDP** for some reason, then unauthorized  
3803 **access** may result. In some environments, the **subject** may be able to suppress the publication of  
3804 **attributes** by the application of privacy controls, or the server or repository that contains the information  
3805 may be unavailable for accidental or intentional reasons.

3806 An example of a changing base group would be if there is a **policy** that everyone in the engineering  
3807 department may change software source code, except for secretaries. Suppose now that the department  
3808 was to merge with another engineering department and the intent is to maintain the same **policy**.  
3809 However, the new department also includes individuals identified as administrative assistants, who ought  
3810 to be treated in the same way as secretaries. Unless the **policy** is altered, they will unintentionally be  
3811 permitted to change software source code. Problems of this type are easy to avoid when one individual  
3812 administers all **policies**, but when administration is distributed, as XACML allows, this type of situation  
3813 must be explicitly guarded against.

## 3814 9.1.8 Denial of service

3815 A denial of service attack is one in which the adversary overloads an XACML actor with excessive  
3816 computations or network traffic such that legitimate users cannot access the services provided by the  
3817 actor.

3818 The urn:oasis:names:tc:xacml:3.0:function:access-permitted function may lead to hard to predict behavior  
3819 in the **PDP**. It is possible that the function is invoked during the recursive invocations of the **PDP** such that  
3820 loops are formed. Such loops may in some cases lead to large numbers of requests to be generated  
3821 before the **PDP** can detect the loop and abort evaluation. Such loops could cause a denial of service at  
3822 the **PDP**, either because of a malicious **policy** or because of a mistake in a **policy**.

## 3823 9.2 Safeguards

### 3824 9.2.1 Authentication

3825 Authentication provides the means for one party in a transaction to determine the identity of the other  
3826 party in the transaction. Authentication may be in one direction, or it may be bilateral.

3827 Given the sensitive nature of **access control** systems, it is important for a **PEP** to authenticate the  
3828 identity of the **PDP** to which it sends **decision requests**. Otherwise, there is a risk that an adversary  
3829 could provide false or invalid **authorization decisions**, leading to a policy violation.

3830 It is equally important for a **PDP** to authenticate the identity of the **PEP** and assess the level of trust to  
3831 determine what, if any, sensitive data should be passed. One should keep in mind that even simple  
3832 "Permit" or "Deny" responses could be exploited if an adversary were allowed to make unlimited requests  
3833 to a **PDP**.

3834 Many different techniques may be used to provide authentication, such as co-located code, a private  
3835 network, a VPN, or digital signatures. Authentication may also be performed as part of the  
3836 communication protocol used to exchange the **contexts**. In this case, authentication may be performed  
3837 either at the message level or at the session level.

### 3838 9.2.2 Policy administration

3839 If the contents of **policies** are exposed outside of the **access control** system, potential **subjects** may  
3840 use this information to determine how to gain unauthorized **access**.

3841 To prevent this threat, the repository used for the storage of **policies** may itself require **access control**.  
3842 In addition, the <Status> element should be used to return values of missing **attributes** only when  
3843 exposure of the identities of those **attributes** will not compromise security.

## 3844 9.2.3 Confidentiality

3845 Confidentiality mechanisms ensure that the contents of a message can be read only by the desired  
3846 recipients and not by anyone else who encounters the message while it is in transit. There are two areas  
3847 in which confidentiality should be considered: one is confidentiality during transmission; the other is  
3848 confidentiality within a <Policy> element.

### 3849 9.2.3.1 Communication confidentiality

3850 In some environments it is deemed good practice to treat all data within an **access control** system as  
3851 confidential. In other environments, **policies** may be made freely available for distribution, inspection,  
3852 and audit. The idea behind keeping **policy** information secret is to make it more difficult for an adversary  
3853 to know what steps might be sufficient to obtain unauthorized **access**. Regardless of the approach  
3854 chosen, the security of the **access control** system should not depend on the secrecy of the **policy**.

3855 Any security considerations related to transmitting or exchanging XACML <Policy> elements are  
3856 outside the scope of the XACML standard. While it is important to ensure that the integrity and  
3857 confidentiality of <Policy> elements is maintained when they are exchanged between two parties, it is  
3858 left to the implementers to determine the appropriate mechanisms for their environment.

3859 Communications confidentiality can be provided by a confidentiality mechanism, such as SSL. Using a  
3860 point-to-point scheme like SSL may lead to other vulnerabilities when one of the end-points is  
3861 compromised.

### 3862 9.2.3.2 Statement level confidentiality

3863 In some cases, an implementation may want to encrypt only parts of an XACML <Policy> element.

3864 The XML Encryption Syntax and Processing Candidate Recommendation from W3C can be used to  
3865 encrypt all or parts of an XML document. This specification is recommended for use with XACML.

3866 It should go without saying that if a repository is used to facilitate the communication of cleartext (i.e.,  
3867 unencrypted) **policy** between the **PAP** and **PDP**, then a secure repository should be used to store this  
3868 sensitive data.

## 3869 9.2.4 Policy integrity

3870 The XACML **policy** used by the **PDP** to evaluate the request **context** is the heart of the system.  
3871 Therefore, maintaining its integrity is essential. There are two aspects to maintaining the integrity of the  
3872 **policy**. One is to ensure that <Policy> elements have not been altered since they were originally  
3873 created by the **PAP**. The other is to ensure that <Policy> elements have not been inserted or deleted  
3874 from the set of **policies**.

3875 In many cases, both aspects can be achieved by ensuring the integrity of the actors and implementing  
3876 session-level mechanisms to secure the communication between actors. The selection of the appropriate  
3877 mechanisms is left to the implementers. However, when **policy** is distributed between organizations to  
3878 be acted on at a later time, or when the **policy** travels with the protected **resource**, it would be useful to  
3879 sign the **policy**. In these cases, the XML Signature Syntax and Processing standard from W3C is  
3880 recommended to be used with XACML.

3881 Digital signatures should only be used to ensure the integrity of the statements. Digital signatures should  
3882 not be used as a method of selecting or evaluating **policy**. That is, the **PDP** should not request a **policy**  
3883 based on who signed it or whether or not it has been signed (as such a basis for selection would, itself,  
3884 be a matter of policy). However, the **PDP** must verify that the key used to sign the **policy** is one  
3885 controlled by the purported **issuer** of the **policy**. The means to do this are dependent on the specific  
3886 signature technology chosen and are outside the scope of this document.

## 3887 9.2.5 Policy identifiers

3888 Since **policies** can be referenced by their identifiers, it is the responsibility of the **PAP** to ensure that  
3889 these are unique. Confusion between identifiers could lead to misidentification of the **applicable policy**.

3890 This specification is silent on whether a **PAP** must generate a new identifier when a **policy** is modified or  
3891 may use the same identifier in the modified **policy**. This is a matter of administrative practice. However,  
3892 care must be taken in either case. If the identifier is reused, there is a danger that other **policies** or  
3893 **policy sets** that reference it may be adversely affected. Conversely, if a new identifier is used, these  
3894 other **policies** may continue to use the prior **policy**, unless it is deleted. In either case the results may  
3895 not be what the **policy** administrator intends.

3896 If a **PDP** is provided with **policies** from distinct sources which might not be fully trusted, as in the use of  
3897 the administration profile [**XACMLAdmin**], there is a concern that someone could intentionally publish a  
3898 **policy** with an id which collides with another **policy**. This could cause **policy** references that point to the  
3899 wrong **policy**, and may cause other unintended consequences in an implementation which is predicated  
3900 upon having unique **policy** identifiers.

3901 If this issue is a concern it is RECOMMENDED that distinct **policy** issuers or sources are assigned  
3902 distinct namespaces for **policy** identifiers. One method is to make sure that the **policy** identifier begins  
3903 with a string which has been assigned to the particular **policy** issuer or source. The remainder of the  
3904 **policy** identifier is an issuer-specific unique part. For instance, Alice from Example Inc. could be assigned  
3905 the **policy** identifiers which begin with `http://example.com/xacml/policyId/alice/`. The **PDP** or another  
3906 trusted component can then verify that the authenticated source of the **policy** is Alice at Example Inc, or  
3907 otherwise reject the **policy**. Anyone else will be unable to publish **policies** with identifiers which collide  
3908 with the **policies** of Alice.

## 3909 9.2.6 Trust model

3910 Discussions of authentication, integrity and confidentiality safeguards necessarily assume an underlying  
3911 trust model: how can one actor come to believe that a given key is uniquely associated with a specific,  
3912 identified actor so that the key can be used to encrypt data for that actor or verify signatures (or other  
3913 integrity structures) from that actor? Many different types of trust models exist, including strict  
3914 hierarchies, distributed authorities, the Web, the bridge, and so on.

3915 It is worth considering the relationships between the various actors of the **access control** system in terms  
3916 of the interdependencies that do and do not exist.

- 3917 • None of the entities of the authorization system are dependent on the **PEP**. They may collect data  
3918 from it, (for example authentication data) but are responsible for verifying it themselves.
- 3919 • The correct operation of the system depends on the ability of the **PEP** to actually enforce **policy**  
3920 **decisions**.
- 3921 • The **PEP** depends on the **PDP** to correctly evaluate **policies**. This in turn implies that the **PDP** is  
3922 supplied with the correct inputs. Other than that, the **PDP** does not depend on the **PEP**.
- 3923 • The **PDP** depends on the **PAP** to supply appropriate **policies**. The **PAP** is not dependent on other  
3924 components.

## 3925 9.2.7 Privacy

3926 It is important to be aware that any transactions that occur with respect to **access control** may reveal  
3927 private information about the actors. For example, if an XACML **policy** states that certain data may only  
3928 be read by **subjects** with "Gold Card Member" status, then any transaction in which a **subject** is  
3929 permitted **access** to that data leaks information to an adversary about the **subject's** status. Privacy  
3930 considerations may therefore lead to encryption and/or to **access control** requirements surrounding the  
3931 enforcement of XACML **policy** instances themselves: confidentiality-protected channels for the  
3932 request/response protocol messages, protection of **subject attributes** in storage and in transit, and so  
3933 on.

3934 Selection and use of privacy mechanisms appropriate to a given environment are outside the scope of  
3935 XACML. The **decision** regarding whether, how, and when to deploy such mechanisms is left to the  
3936 implementers associated with the environment.

3937 **9.3 Unicode security issues**

3938 There are many security considerations related to use of Unicode. An XACML implementation SHOULD  
3939 follow the advice given in the relevant version of **[UTR36]**.

3940 **9.4 Identifier equality**

3941 Section 7.20 defines the identifier equality operation for XACML. This definition of equality does not do  
3942 any kind of canonicalization or escaping of characters. The identifiers defined in the XACML specification  
3943 have been selected to not include any ambiguity regarding these aspects. It is RECOMMENDED that  
3944 identifiers defined by extensions also do not introduce any identifiers which might be mistaken for being  
3945 subject to processing, like for instance URL character encoding using “%”.

---

## 3946 10 Conformance

### 3947 10.1 Introduction

3948 The XACML specification addresses the following aspect of conformance:

3949 The XACML specification defines a number of functions, etc. that have somewhat special applications,  
3950 therefore they are not required to be implemented in an implementation that claims to conform with the  
3951 OASIS standard.

### 3952 10.2 Conformance tables

3953 This section lists those portions of the specification that **MUST** be included in an implementation of a **PDP**  
3954 that claims to conform to XACML v3.0. A set of test cases has been created to assist in this process.  
3955 These test cases can be located from the OASIS XACML TC Web page. The site hosting the test cases  
3956 contains a full description of the test cases and how to execute them.

3957 Note: "M" means mandatory-to-implement. "O" means optional.

3958 The implementation **MUST** follow sections 5, 6, 7, Appendix A, Appendix B and Appendix C where they  
3959 apply to implemented items in the following tables.

#### 3960 10.2.1 Schema elements

3961 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

3962 **10.2.2 Identifier Prefixes**

3963 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

3964 **10.2.3 Algorithms**

3965 The implementation MUST include the **rule-** and **policy-combining algorithms** associated with the  
 3966 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

## 3967 10.2.4 Status Codes

3968 Implementation support for the <StatusCode> element is optional, but if the element is supported, then  
3969 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

## 3970 10.2.5 Attributes

3971 The implementation MUST support the **attributes** associated with the following identifiers as specified by  
3972 XACML. If values for these **attributes** are not present in the **decision request**, then their values MUST  
3973 be supplied by the **context handler**. So, unlike most other **attributes**, their semantics are not  
3974 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

## 3975 10.2.6 Identifiers

3976 The implementation MUST use the **attributes** associated with the following identifiers in the way XACML  
3977 has defined. This requirement pertains primarily to implementations of a **PAP** or **PEP** that uses XACML,  
3978 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:2.0:resource:target-namespace	O
urn:oasis:names:tc:xacml:1.0:action:action-id	
urn:oasis:names:tc:xacml:1.0:action:action-namespace	O
urn:oasis:names:tc:xacml:1.0:action:implied-action	

3979 **10.2.7 Data-types**

3980 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

3981 **10.2.8 Functions**

3982 The implementation MUST properly process those functions associated with the identifiers marked with  
3983 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

3984 **10.2.9 Identifiers planned for future deprecation**

3985 These identifiers are associated with previous versions of XACML and newer alternatives exist in XACML  
3986 3.0. They are planned to be deprecated at some unspecified point in the future. It is RECOMMENDED  
3987 that these identifiers not be used in new policies and requests.

3988 The implementation MUST properly process those features associated with the identifiers marked with an  
3989 "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

3990



3991  
3992  
3993  
3994  
3995  
3996  
3997  
  
3998  
  
3999  
4000  
4001  
4002  
4003  
4004  
4005  
4006  
4007  
4008  
4009  
4010  
4011  
4012  
4013  
4014  
4015  
4016  
4017  
4018  
4019  
4020  
4021  
4022  
4023  
4024  
4025  
4026  
4027  
4028  
4029  
4030

---

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

### A.1 Introduction

This section specifies the data-types and functions used in XACML to create *predicates* for *conditions* and *target* matches.

This specification combines the various standards set forth by IEEE and ANSI for string representation of numeric values, as well as the evaluation of arithmetic functions. It describes the primitive data-types and *bags*. The standard functions are named and their operational semantics are described.

### A.2 Data-types

Although XML instances represent all data-types as strings, an XACML *PDP* must operate on types of data that, while they have string representations, are not just strings. Types such as Boolean, integer, and double MUST be converted from their XML string representations to values that can be compared with values in their domain of discourse, such as numbers. The following primitive data-types are specified for use with XACML and have explicit data representations:

- <http://www.w3.org/2001/XMLSchema#string>
- <http://www.w3.org/2001/XMLSchema#boolean>
- <http://www.w3.org/2001/XMLSchema#integer>
- <http://www.w3.org/2001/XMLSchema#double>
- <http://www.w3.org/2001/XMLSchema#time>
- <http://www.w3.org/2001/XMLSchema#date>
- <http://www.w3.org/2001/XMLSchema#dateTime>
- <http://www.w3.org/2001/XMLSchema#anyURI>
- <http://www.w3.org/2001/XMLSchema#hexBinary>
- <http://www.w3.org/2001/XMLSchema#base64Binary>
- <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- <http://www.w3.org/2001/XMLSchema#yearMonthDuration>
- <urn:oasis:names:tc:xacml:1.0:data-type:x500Name>
- <urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name>
- <urn:oasis:names:tc:xacml:2.0:data-type:ipAddress>
- <urn:oasis:names:tc:xacml:2.0:data-type:dnsName>
- <urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression>

For the sake of improved interoperability, it is RECOMMENDED that all time references be in UTC time.

An XACML *PDP* SHALL be capable of converting string representations into various primitive data-types. For doubles, XACML SHALL use the conversions described in [IEEE754].

XACML defines four data-types representing identifiers for *subjects* or *resources*; these are:

- “urn:oasis:names:tc:xacml:1.0:data-type:x500Name”,
- “urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name”
- “urn:oasis:names:tc:xacml:2.0:data-type:ipAddress” and
- “urn:oasis:names:tc:xacml:2.0:data-type:dnsName”

These types appear in several standard applications, such as TLS/SSL and electronic mail.

#### X.500 directory name

4031 The "urn:oasis:names:tc:xacml:1.0:data-type:x500Name" primitive type represents an ITU-T Rec.  
4032 X.520 Distinguished Name. The valid syntax for such a name is described in IETF RFC 2253  
4033 "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished  
4034 Names".

#### 4035 **RFC 822 name**

4036 The "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" primitive type represents an electronic  
4037 mail address. The valid syntax for such a name is described in IETF RFC 2821, Section 4.1.2,  
4038 Command Argument Syntax, under the term "Mailbox".

#### 4039 **IP address**

4040 The "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" primitive type represents an IPv4 or IPv6  
4041 network address, with optional mask and optional port or port range. The syntax SHALL be:

4042 ipAddress = address [ "/" mask ] [ ":" [ portrange ] ]

4043 For an IPv4 address, the address and mask are formatted in accordance with the syntax for a  
4044 "host" in IETF RFC 2396 "Uniform Resource Identifiers (URI): Generic Syntax", section 3.2.

4045 For an IPv6 address, the address and mask are formatted in accordance with the syntax for an  
4046 "ipv6reference" in IETF RFC 2732 "Format for Literal IPv6 Addresses in URL's". (Note that an  
4047 IPv6 address or mask, in this syntax, is enclosed in literal "[" "]" brackets.)

#### 4048 **DNS name**

4049 The "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" primitive type represents a Domain  
4050 Name Service (DNS) host name, with optional port or port range. The syntax SHALL be:

4051 dnsName = hostname [ ":" portrange ]

4052 The hostname is formatted in accordance with IETF RFC 2396 "Uniform Resource Identifiers  
4053 (URI): Generic Syntax", section 3.2, except that a wildcard "\*" may be used in the left-most  
4054 component of the hostname to indicate "any subdomain" under the domain specified to its right.

4055 For both the "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" and  
4056 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" data-types, the port or port range syntax  
4057 SHALL be

4058 portrange = portnumber | "-"portnumber | portnumber "-"portnumber

4059 where "portnumber" is a decimal port number. If the port number is of the form "-x", where "x" is  
4060 a port number, then the range is all ports numbered "x" and below. If the port number is of the  
4061 form "x-", then the range is all ports numbered "x" and above. [This syntax is taken from the Java  
4062 SocketPermission.]

#### 4063 **XPath expression**

4064 The "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" primitive type represents an  
4065 XPath expression over the XML in a <Content> element. The syntax is defined by the XPath  
4066 W3C recommendation. The content of this data type also includes the context in which  
4067 namespaces prefixes in the expression are resolved, which distinguishes it from a plain string and  
4068 the XACML **attribute** category of the <Content> element to which it applies. When the value is  
4069 encoded in an <AttributeValue> element, the namespace context is given by the [in-scope  
4070 namespaces] (see [INFOSET]) of the <AttributeValue> element, and an XML attribute called  
4071 XPathCategory gives the category of the <Content> element where the expression applies.

4072 The XPath expression MUST be evaluated in a context which is equivalent of a stand alone XML  
4073 document with the only child of the <Content> element as the document element. Namespace  
4074 declarations which are not "visibly utilized", as defined by [exc-c14n], MAY not be present and  
4075 MUST NOT be utilized by the XPath expression. The context node of the XPath expression is the  
4076 document node of this stand alone document.

## 4077 A.3 Functions

4078 XACML specifies the following functions. Unless otherwise specified, if an argument of one of these  
4079 functions were to evaluate to "Indeterminate", then the function SHALL be set to "Indeterminate".

4080 Note that in each case an implementation is conformant as long as it produces the same result as is  
4081 specified here, regardless of how and in what order the implementation behaves internally.

### 4082 A.3.1 Equality predicates

4083 The following functions are the equality functions for the various primitive types. Each function for a  
4084 particular data-type follows a specified standard convention for that data-type.

- 4085 • urn:oasis:names:tc:xacml:1.0:function:string-equal

4086 This function SHALL take two arguments of data-type  
4087 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an  
4088 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if  
4089 the value of both of its arguments are of equal length and each string is determined to be equal.  
4090 Otherwise, it SHALL return "False". The comparison SHALL use Unicode codepoint collation, as  
4091 defined for the identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.

- 4092 • urn:oasis:names:tc:xacml:3.0:function:string-equal-ignore-case

4093 This function SHALL take two arguments of data-type  
4094 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an  
4095 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be "True" if and only if the  
4096 two strings are equal as defined by urn:oasis:names:tc:xacml:1.0:function:string-equal after they  
4097 have both been converted to lower case with urn:oasis:names:tc:xacml:1.0:function:string-  
4098 normalize-to-lower-case.

- 4099 • urn:oasis:names:tc:xacml:1.0:function:boolean-equal

4100 This function SHALL take two arguments of data-type  
4101 "http://www.w3.org/2001/XMLSchema#boolean" and SHALL return an  
4102 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if  
4103 the arguments are equal. Otherwise, it SHALL return "False".

- 4104 • urn:oasis:names:tc:xacml:1.0:function:integer-equal

4105 This function SHALL take two arguments of data-type  
4106 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return an  
4107 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if  
4108 the two arguments represent the same number.

- 4109 • urn:oasis:names:tc:xacml:1.0:function:double-equal

4110 This function SHALL take two arguments of data-type  
4111 "http://www.w3.org/2001/XMLSchema#double" and SHALL return an  
4112 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation on doubles  
4113 according to IEEE 754 **[IEEE754]**.

- 4114 • urn:oasis:names:tc:xacml:1.0:function:date-equal

4115 This function SHALL take two arguments of data-type  
4116 "http://www.w3.org/2001/XMLSchema#date" and SHALL return an  
4117 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to  
4118 the "op:date-equal" function **[XF]** Section 10.4.9.

- 4119 • urn:oasis:names:tc:xacml:1.0:function:time-equal

4120 This function SHALL take two arguments of data-type  
4121 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an  
4122 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to  
4123 the "op:time-equal" function **[XF]** Section 10.4.12.

- 4124 • urn:oasis:names:tc:xacml:1.0:function:dateTime-equal  
 4125 This function SHALL take two arguments of data-type  
 4126 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an  
 4127 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to  
 4128 the "op:dateTime-equal" function [XF] Section 10.4.6.
- 4129 • urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-equal  
 4130 This function SHALL take two arguments of data-type  
 4131 "http://www.w3.org/2001/XMLSchema#dayTimeDuration" and SHALL return an  
 4132 "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation  
 4133 according to the "op:duration-equal" function [XF] Section 10.4.5. Note that the lexical  
 4134 representation of each argument MUST be converted to a value expressed in fractional seconds  
 4135 [XF] Section 10.3.2.
- 4136 • urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-equal  
 4137 This function SHALL take two arguments of data-type  
 4138 "http://www.w3.org/2001/XMLSchema#yearMonthDuration" and SHALL return an  
 4139 "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation  
 4140 according to the "op:duration-equal" function [XF] Section 10.4.5. Note that the lexical  
 4141 representation of each argument MUST be converted to a value expressed in fractional seconds  
 4142 [XF] Section 10.3.2.
- 4143 • urn:oasis:names:tc:xacml:1.0:function:anyURI-equal  
 4144 This function SHALL take two arguments of data-type  
 4145 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return an  
 4146 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL convert the arguments to  
 4147 strings with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI and return "True" if and  
 4148 only if the values of the two arguments are equal on a codepoint-by-codepoint basis.
- 4149 • urn:oasis:names:tc:xacml:1.0:function:x500Name-equal  
 4150 This function SHALL take two arguments of "urn:oasis:names:tc:xacml:1.0:data-type:x500Name"  
 4151 and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if  
 4152 and only if each Relative Distinguished Name (RDN) in the two arguments matches. Otherwise,  
 4153 it SHALL return "False". Two RDNs shall be said to match if and only if the result of the following  
 4154 operations is "True" .
- 4155 1. Normalize the two arguments according to IETF RFC 2253 "Lightweight Directory Access  
 4156 Protocol (v3): UTF-8 String Representation of Distinguished Names".
  - 4157 2. If any RDN contains multiple attributeTypeAndValue pairs, re-order the Attribute  
 4158 ValuePairs in that RDN in ascending order when compared as octet strings (described in  
 4159 ITU-T Rec. X.690 (1997 E) Section 11.6 "Set-of components").
  - 4160 3. Compare RDNs using the rules in IETF RFC 3280 "Internet X.509 Public Key  
 4161 Infrastructure Certificate and Certificate Revocation List (CRL) Profile", Section 4.1.2.4  
 4162 "Issuer".
- 4163 • urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal  
 4164 This function SHALL take two arguments of data-type "urn:oasis:names:tc:xacml:1.0:data-  
 4165 type:rfc822Name" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It  
 4166 SHALL return "True" if and only if the two arguments are equal. Otherwise, it SHALL return  
 4167 "False". An RFC822 name consists of a local-part followed by "@" followed by a domain-part.  
 4168 The local-part is case-sensitive, while the domain-part (which is usually a DNS host name) is not  
 4169 case-sensitive. Perform the following operations:
- 4170 1. Normalize the domain-part of each argument to lower case
  - 4171 2. Compare the expressions by applying the function  
 4172 "urn:oasis:names:tc:xacml:1.0:function:string-equal" to the normalized arguments.

4173 • urn:oasis:names:tc:xacml:1.0:function:hexBinary-equal  
4174 This function SHALL take two arguments of data-type  
4175 "http://www.w3.org/2001/XMLSchema#hexBinary" and SHALL return an  
4176 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet sequences  
4177 represented by the value of both arguments have equal length and are equal in a conjunctive,  
4178 point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function.  
4179 Otherwise, it SHALL return "False". The conversion from the string representation to an octet  
4180 sequence SHALL be as specified in [XS] Section 3.2.15.

4181 • urn:oasis:names:tc:xacml:1.0:function:base64Binary-equal  
4182 This function SHALL take two arguments of data-type  
4183 "http://www.w3.org/2001/XMLSchema#base64Binary" and SHALL return an  
4184 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet sequences  
4185 represented by the value of both arguments have equal length and are equal in a conjunctive,  
4186 point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function.  
4187 Otherwise, it SHALL return "False". The conversion from the string representation to an octet  
4188 sequence SHALL be as specified in [XS] Section 3.2.16.

### 4189 A.3.2 Arithmetic functions

4190 All of the following functions SHALL take two arguments of the specified data-type, integer, or double,  
4191 and SHALL return an element of integer or double data-type, respectively. However, the "add" and  
4192 "multiply" functions MAY take more than two arguments. Each function evaluation operating on doubles  
4193 SHALL proceed as specified by their logical counterparts in IEEE 754 [IEEE754]. For all of these  
4194 functions, if any argument is "Indeterminate", then the function SHALL evaluate to "Indeterminate". In the  
4195 case of the divide functions, if the divisor is zero, then the function SHALL evaluate to "Indeterminate".

4196 • urn:oasis:names:tc:xacml:1.0:function:integer-add  
4197 This function MUST accept two or more arguments.

4198 • urn:oasis:names:tc:xacml:1.0:function:double-add  
4199 This function MUST accept two or more arguments.

4200 • urn:oasis:names:tc:xacml:1.0:function:integer-subtract  
4201 The result is the second argument subtracted from the first argument.

4202 • urn:oasis:names:tc:xacml:1.0:function:double-subtract  
4203 The result is the second argument subtracted from the first argument.

4204 • urn:oasis:names:tc:xacml:1.0:function:integer-multiply  
4205 This function MUST accept two or more arguments.

4206 • urn:oasis:names:tc:xacml:1.0:function:double-multiply  
4207 This function MUST accept two or more arguments.

4208 • urn:oasis:names:tc:xacml:1.0:function:integer-divide  
4209 The result is the first argument divided by the second argument.

4210 • urn:oasis:names:tc:xacml:1.0:function:double-divide  
4211 The result is the first argument divided by the second argument.

4212 • urn:oasis:names:tc:xacml:1.0:function:integer-mod  
4213 The result is remainder of the first argument divided by the second argument.

4214 The following functions SHALL take a single argument of the specified data-type. The round and floor  
4215 functions SHALL take a single argument of data-type "http://www.w3.org/2001/XMLSchema#double" and  
4216 return a value of the data-type "http://www.w3.org/2001/XMLSchema#double".

4217 • urn:oasis:names:tc:xacml:1.0:function:integer-abs

- 4218 • urn:oasis:names:tc:xacml:1.0:function:double-abs
- 4219 • urn:oasis:names:tc:xacml:1.0:function:round
- 4220 • urn:oasis:names:tc:xacml:1.0:function:floor

### 4221 **A.3.3 String conversion functions**

4222 The following functions convert between values of the data-type  
4223 “http://www.w3.org/2001/XMLSchema#string” primitive types.

- 4224 • urn:oasis:names:tc:xacml:1.0:function:string-normalize-space
  - 4225 This function SHALL take one argument of data-type
  - 4226 “http://www.w3.org/2001/XMLSchema#string” and SHALL normalize the value by stripping off all
  - 4227 leading and trailing white space characters. The whitespace characters are defined in the
  - 4228 metasympol S (Production 3) of [XML].
- 4229 • urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case
  - 4230 This function SHALL take one argument of data-type
  - 4231 “http://www.w3.org/2001/XMLSchema#string” and SHALL normalize the value by converting each
  - 4232 upper case character to its lower case equivalent. Case mapping shall be done as specified for
  - 4233 the fn:lower-case function in [XF] with no tailoring for particular languages or environments.

### 4234 **A.3.4 Numeric data-type conversion functions**

4235 The following functions convert between the data-type “http://www.w3.org/2001/XMLSchema#integer”  
4236 and” http://www.w3.org/2001/XMLSchema#double” primitive types.

- 4237 • urn:oasis:names:tc:xacml:1.0:function:double-to-integer
  - 4238 This function SHALL take one argument of data-type
  - 4239 “http://www.w3.org/2001/XMLSchema#double” and SHALL truncate its numeric value to a whole
  - 4240 number and return an element of data-type “http://www.w3.org/2001/XMLSchema#integer”.
- 4241 • urn:oasis:names:tc:xacml:1.0:function:integer-to-double
  - 4242 This function SHALL take one argument of data-type
  - 4243 “http://www.w3.org/2001/XMLSchema#integer” and SHALL promote its value to an element of
  - 4244 data-type “http://www.w3.org/2001/XMLSchema#double” with the same numeric value. If the
  - 4245 integer argument is outside the range which can be represented by a double, the result SHALL
  - 4246 be Indeterminate, with the status code “urn:oasis:names:tc:xacml:1.0:status:processing-error”.

### 4247 **A.3.5 Logical functions**

4248 This section contains the specification for logical functions that operate on arguments of data-type  
4249 “http://www.w3.org/2001/XMLSchema#boolean”.

- 4250 • urn:oasis:names:tc:xacml:1.0:function:or
  - 4251 This function SHALL return "False" if it has no arguments and SHALL return "True" if at least one
  - 4252 of its arguments evaluates to "True". The order of evaluation SHALL be from first argument to
  - 4253 last. The evaluation SHALL stop with a result of "True" if any argument evaluates to "True",
  - 4254 leaving the rest of the arguments unevaluated.
- 4255 • urn:oasis:names:tc:xacml:1.0:function:and
  - 4256 This function SHALL return "True" if it has no arguments and SHALL return "False" if one of its
  - 4257 arguments evaluates to "False". The order of evaluation SHALL be from first argument to last.
  - 4258 The evaluation SHALL stop with a result of "False" if any argument evaluates to "False", leaving
  - 4259 the rest of the arguments unevaluated.
- 4260 • urn:oasis:names:tc:xacml:1.0:function:n-of
  - 4261 The first argument to this function SHALL be of data-type
  - 4262 http://www.w3.org/2001/XMLSchema#integer. The remaining arguments SHALL be of data-type



4263 http://www.w3.org/2001/XMLSchema#boolean. The first argument specifies the minimum  
4264 number of the remaining arguments that MUST evaluate to "True" for the expression to be  
4265 considered "True". If the first argument is 0, the result SHALL be "True". If the number of  
4266 arguments after the first one is less than the value of the first argument, then the expression  
4267 SHALL result in "Indeterminate". The order of evaluation SHALL be: first evaluate the integer  
4268 value, and then evaluate each subsequent argument. The evaluation SHALL stop and return  
4269 "True" if the specified number of arguments evaluate to "True". The evaluation of arguments  
4270 SHALL stop if it is determined that evaluating the remaining arguments will not satisfy the  
4271 requirement.

4272 • urn:oasis:names:tc:xacml:1.0:function:not

4273 This function SHALL take one argument of data-type  
4274 "http://www.w3.org/2001/XMLSchema#boolean". If the argument evaluates to "True", then the  
4275 result of the expression SHALL be "False". If the argument evaluates to "False", then the result  
4276 of the expression SHALL be "True".

4277 Note: When evaluating and, or, or n-of, it may not be necessary to attempt a full evaluation of each  
4278 argument in order to determine whether the evaluation of the argument would result in "Indeterminate".  
4279 Analysis of the argument regarding the availability of its **attributes**, or other analysis regarding errors,  
4280 such as "divide-by-zero", may render the argument error free. Such arguments occurring in the  
4281 expression in a position after the evaluation is stated to stop need not be processed.

### 4282 A.3.6 Numeric comparison functions

4283 These functions form a minimal set for comparing two numbers, yielding a Boolean result. For doubles  
4284 they SHALL comply with the rules governed by IEEE 754 [IEEE754].

- 4285 • urn:oasis:names:tc:xacml:1.0:function:integer-greater-than
- 4286 • urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal
- 4287 • urn:oasis:names:tc:xacml:1.0:function:integer-less-than
- 4288 • urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal
- 4289 • urn:oasis:names:tc:xacml:1.0:function:double-greater-than
- 4290 • urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal
- 4291 • urn:oasis:names:tc:xacml:1.0:function:double-less-than
- 4292 • urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal

### 4293 A.3.7 Date and time arithmetic functions

4294 These functions perform arithmetic operations with date and time.

4295 • urn:oasis:names:tc:xacml:3.0:function:dateTime-add-dayTimeDuration

4296 This function SHALL take two arguments, the first SHALL be of data-type  
4297 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be of data-type  
4298 "http://www.w3.org/2001/XMLSchema#dayTimeDuration". It SHALL return a result of  
4299 "http://www.w3.org/2001/XMLSchema#dateTime". This function SHALL return the value by  
4300 adding the second argument to the first argument according to the specification of adding  
4301 durations to date and time [XS] Appendix E.

4302 • urn:oasis:names:tc:xacml:3.0:function:dateTime-add-yearMonthDuration

4303 This function SHALL take two arguments, the first SHALL be a  
4304 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a  
4305 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of  
4306 "http://www.w3.org/2001/XMLSchema#dateTime". This function SHALL return the value by  
4307 adding the second argument to the first argument according to the specification of adding  
4308 durations to date and time [XS] Appendix E.



- 4309 • urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-dayTimeDuration
  - 4310 This function SHALL take two arguments, the first SHALL be a
  - 4311 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a
  - 4312 "http://www.w3.org/2001/XMLSchema#dayTimeDuration". It SHALL return a result of
  - 4313 "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration,
  - 4314 then this function SHALL return the value by adding the corresponding negative duration, as per
  - 4315 the specification [XS] Appendix E. If the second argument is a negative duration, then the result
  - 4316 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-
  - 4317 dayTimeDuration" had been applied to the corresponding positive duration.
- 4318 • urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-yearMonthDuration
  - 4319 This function SHALL take two arguments, the first SHALL be a
  - 4320 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a
  - 4321 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
  - 4322 "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration,
  - 4323 then this function SHALL return the value by adding the corresponding negative duration, as per
  - 4324 the specification [XS] Appendix E. If the second argument is a negative duration, then the result
  - 4325 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-
  - 4326 yearMonthDuration" had been applied to the corresponding positive duration.
- 4327 • urn:oasis:names:tc:xacml:3.0:function:date-add-yearMonthDuration
  - 4328 This function SHALL take two arguments, the first SHALL be a
  - 4329 "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a
  - 4330 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
  - 4331 "http://www.w3.org/2001/XMLSchema#date". This function SHALL return the value by adding the
  - 4332 second argument to the first argument according to the specification of adding duration to date
  - 4333 [XS] Appendix E.
- 4334 • urn:oasis:names:tc:xacml:3.0:function:date-subtract-yearMonthDuration
  - 4335 This function SHALL take two arguments, the first SHALL be a
  - 4336 "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a
  - 4337 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
  - 4338 "http://www.w3.org/2001/XMLSchema#date". If the second argument is a positive duration, then
  - 4339 this function SHALL return the value by adding the corresponding negative duration, as per the
  - 4340 specification [XS] Appendix E. If the second argument is a negative duration, then the result
  - 4341 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration"
  - 4342 had been applied to the corresponding positive duration.

### 4343 A.3.8 Non-numeric comparison functions

4344 These functions perform comparison operations on two arguments of non-numerical types.

- 4345 • urn:oasis:names:tc:xacml:1.0:function:string-greater-than
  - 4346 This function SHALL take two arguments of data-type
  - 4347 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
  - 4348 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
  - 4349 argument is lexicographically strictly greater than the second argument. Otherwise, it SHALL
  - 4350 return "False". The comparison SHALL use Unicode codepoint collation, as defined for the
  - 4351 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [XF].
- 4352 • urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal
  - 4353 This function SHALL take two arguments of data-type
  - 4354 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
  - 4355 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
  - 4356 argument is lexicographically greater than or equal to the second argument. Otherwise, it SHALL
  - 4357 return "False". The comparison SHALL use Unicode codepoint collation, as defined for the
  - 4358 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [XF].

- 4359 • urn:oasis:names:tc:xacml:1.0:function:string-less-than
  - 4360 This function SHALL take two arguments of data-type
  - 4361 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
  - 4362 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only the first
  - 4363 argument is lexicographically strictly less than the second argument. Otherwise, it SHALL return
  - 4364 "False". The comparison SHALL use Unicode codepoint collation, as defined for the identifier
  - 4365 http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.
- 4366 • urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal
  - 4367 This function SHALL take two arguments of data-type
  - 4368 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
  - 4369 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only the first
  - 4370 argument is lexicographically less than or equal to the second argument. Otherwise, it SHALL
  - 4371 return "False". The comparison SHALL use Unicode codepoint collation, as defined for the
  - 4372 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.
- 4373 • urn:oasis:names:tc:xacml:1.0:function:time-greater-than
  - 4374 This function SHALL take two arguments of data-type
  - 4375 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
  - 4376 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
  - 4377 argument is greater than the second argument according to the order relation specified for
  - 4378 "http://www.w3.org/2001/XMLSchema#time" **[XS]** Section 3.2.8. Otherwise, it SHALL return
  - 4379 "False". Note: it is illegal to compare a time that includes a time-zone value with one that does
  - 4380 not. In such cases, the time-in-range function should be used.
- 4381 • urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal
  - 4382 This function SHALL take two arguments of data-type
  - 4383 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
  - 4384 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
  - 4385 argument is greater than or equal to the second argument according to the order relation
  - 4386 specified for "http://www.w3.org/2001/XMLSchema#time" **[XS]** Section 3.2.8. Otherwise, it
  - 4387 SHALL return "False". Note: it is illegal to compare a time that includes a time-zone value with
  - 4388 one that does not. In such cases, the time-in-range function should be used.
- 4389 • urn:oasis:names:tc:xacml:1.0:function:time-less-than
  - 4390 This function SHALL take two arguments of data-type
  - 4391 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
  - 4392 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
  - 4393 argument is less than the second argument according to the order relation specified for
  - 4394 "http://www.w3.org/2001/XMLSchema#time" **[XS]** Section 3.2.8. Otherwise, it SHALL return
  - 4395 "False". Note: it is illegal to compare a time that includes a time-zone value with one that does
  - 4396 not. In such cases, the time-in-range function should be used.
- 4397 • urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal
  - 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 less than or equal to the second argument according to the order relation specified
  - 4402 for "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:2.0:function:time-in-range
  - 4406 This function SHALL take three 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 the first argument falls
  - 4409 in the range defined inclusively by the second and third arguments. Otherwise, it SHALL return

4410 “False”. Regardless of its value, the third argument SHALL be interpreted as a time that is equal  
4411 to, or later than by less than twenty-four hours, the second argument. If no time zone is provided  
4412 for the first argument, it SHALL use the default time zone at the **context handler**. If no time zone  
4413 is provided for the second or third arguments, then they SHALL use the time zone from the first  
4414 argument.

- 4415 • urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than

4416 This function SHALL take two arguments of data-type  
4417 “http://www.w3.org/2001/XMLSchema#dateTime” and SHALL return an  
4418 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first  
4419 argument is greater than the second argument according to the order relation specified for  
4420 “http://www.w3.org/2001/XMLSchema#dateTime” by [XS] part 2, section 3.2.7. Otherwise, it  
4421 SHALL return “False”. Note: if a dateTime value does not include a time-zone value, then an  
4422 implicit time-zone value SHALL be assigned, as described in [XS].

- 4423 • urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal

4424 This function SHALL take two arguments of data-type  
4425 “http://www.w3.org/2001/XMLSchema#dateTime” and SHALL return an  
4426 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first  
4427 argument is greater than or equal to the second argument according to the order relation  
4428 specified for “http://www.w3.org/2001/XMLSchema#dateTime” by [XS] part 2, section 3.2.7.  
4429 Otherwise, it SHALL return “False”. Note: if a dateTime value does not include a time-zone  
4430 value, then an implicit time-zone value SHALL be assigned, as described in [XS].

- 4431 • urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than

4432 This function SHALL take two arguments of data-type  
4433 “http://www.w3.org/2001/XMLSchema#dateTime” and SHALL return an  
4434 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first  
4435 argument is less than the second argument according to the order relation specified for  
4436 “http://www.w3.org/2001/XMLSchema#dateTime” by [XS, part 2, section 3.2.7]. Otherwise, it  
4437 SHALL return “False”. Note: if a dateTime value does not include a time-zone value, then an  
4438 implicit time-zone value SHALL be assigned, as described in [XS].

- 4439 • urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal

4440 This function SHALL take two arguments of data-type “http://www.w3.org/2001/XMLSchema#  
4441 dateTime” and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL  
4442 return “True” if and only if the first argument is less than or equal to the second argument  
4443 according to the order relation specified for “http://www.w3.org/2001/XMLSchema#dateTime” by  
4444 [XS] part 2, section 3.2.7. Otherwise, it SHALL return “False”. Note: if a dateTime value does  
4445 not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described  
4446 in [XS].

- 4447 • urn:oasis:names:tc:xacml:1.0:function:date-greater-than

4448 This function SHALL take two arguments of data-type  
4449 “http://www.w3.org/2001/XMLSchema#date” 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 the second argument according to the order relation specified for  
4452 “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9. Otherwise, it SHALL  
4453 return “False”. Note: if a date value does not include a time-zone value, then an implicit time-  
4454 zone value SHALL be assigned, as described in [XS].

- 4455 • urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal

4456 This function SHALL take two arguments of data-type  
4457 “http://www.w3.org/2001/XMLSchema#date” 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 greater than or equal to the second argument according to the order relation  
4460 specified for “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9.

- 4461            Otherwise, it SHALL return “False”. Note: if a date value does not include a time-zone value,  
4462            then an implicit time-zone value SHALL be assigned, as described in [XS].
- 4463            • urn:oasis:names:tc:xacml:1.0:function:date-less-than
    - 4464                This function SHALL take two arguments of data-type
    - 4465                “http://www.w3.org/2001/XMLSchema#date” and SHALL return an
    - 4466                “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
    - 4467                argument is less than the second argument according to the order relation specified for
    - 4468                “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9. Otherwise, it SHALL
    - 4469                return “False”. Note: if a date value does not include a time-zone value, then an implicit time-
    - 4470                zone value SHALL be assigned, as described in [XS].
  - 4471            • urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal
    - 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 less than or equal to the second argument according to the order relation specified
    - 4476                for “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9. Otherwise, it
    - 4477                SHALL return “False”. Note: if a date value does not include a time-zone value, then an implicit
    - 4478                time-zone value SHALL be assigned, as described in [XS].

### 4479    **A.3.9 String functions**

4480    The following functions operate on strings and convert to and from other data types.

- 4481            • urn:oasis:names:tc:xacml:2.0:function:string-concatenate
  - 4482                This function SHALL take two or more arguments of data-type
  - 4483                “http://www.w3.org/2001/XMLSchema#string” and SHALL return a
  - 4484                “http://www.w3.org/2001/XMLSchema#string”. The result SHALL be the concatenation, in order,
  - 4485                of the arguments.
- 4486            • urn:oasis:names:tc:xacml:3.0:function:boolean-from-string
  - 4487                This function SHALL take one argument of data-type
  - 4488                “http://www.w3.org/2001/XMLSchema#string”, and SHALL return an
  - 4489                “http://www.w3.org/2001/XMLSchema#boolean”. The result SHALL be the string converted to a
  - 4490                boolean. If the argument is not a valid lexical representation of a boolean, then the result SHALL
  - 4491                be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4492            • urn:oasis:names:tc:xacml:3.0:function:string-from-boolean
  - 4493                This function SHALL take one argument of data-type
  - 4494                “http://www.w3.org/2001/XMLSchema#boolean”, and SHALL return an
  - 4495                “http://www.w3.org/2001/XMLSchema#string”. The result SHALL be the boolean converted to a
  - 4496                string in the canonical form specified in [XS].
- 4497            • urn:oasis:names:tc:xacml:3.0:function:integer-from-string
  - 4498                This function SHALL take one argument of data-type
  - 4499                “http://www.w3.org/2001/XMLSchema#string”, and SHALL return an
  - 4500                “http://www.w3.org/2001/XMLSchema#integer”. The result SHALL be the string converted to an
  - 4501                integer. If the argument is not a valid lexical representation of an integer, then the result SHALL
  - 4502                be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4503            • urn:oasis:names:tc:xacml:3.0:function:string-from-integer
  - 4504                This function SHALL take one argument of data-type
  - 4505                “http://www.w3.org/2001/XMLSchema#integer”, and SHALL return an
  - 4506                “http://www.w3.org/2001/XMLSchema#string”. The result SHALL be the integer converted to a
  - 4507                string in the canonical form specified in [XS].
- 4508            • urn:oasis:names:tc:xacml:3.0:function:double-from-string

- 4509 This function SHALL take one argument of data-type  
 4510 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an  
 4511 "http://www.w3.org/2001/XMLSchema#double". The result SHALL be the string converted to a  
 4512 double. If the argument is not a valid lexical representation of a double, then the result SHALL be  
 4513 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4514 • urn:oasis:names:tc:xacml:3.0:function:string-from-double
 

4515 This function SHALL take one argument of data-type  
 4516 "http://www.w3.org/2001/XMLSchema#double", and SHALL return an  
 4517 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the double converted to a  
 4518 string in the canonical form specified in **[XS]**.
  - 4519 • urn:oasis:names:tc:xacml:3.0:function:time-from-string
 

4520 This function SHALL take one argument of data-type  
 4521 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an  
 4522 "http://www.w3.org/2001/XMLSchema#time". The result SHALL be the string converted to a time.  
 4523 If the argument is not a valid lexical representation of a time, then the result SHALL be  
 4524 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
  - 4525 • urn:oasis:names:tc:xacml:3.0:function:string-from-time
 

4526 This function SHALL take one argument of data-type  
 4527 "http://www.w3.org/2001/XMLSchema#time", and SHALL return an  
 4528 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the time converted to a  
 4529 string in the canonical form specified in **[XS]**.
  - 4530 • urn:oasis:names:tc:xacml:3.0:function:date-from-string
 

4531 This function SHALL take one argument of data-type  
 4532 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an  
 4533 "http://www.w3.org/2001/XMLSchema#date". The result SHALL be the string converted to a  
 4534 date. If the argument is not a valid lexical representation of a date, then the result SHALL be  
 4535 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
  - 4536 • urn:oasis:names:tc:xacml:3.0:function:string-from-date
 

4537 This function SHALL take one argument of data-type  
 4538 "http://www.w3.org/2001/XMLSchema#date", and SHALL return an  
 4539 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the date converted to a  
 4540 string in the canonical form specified in **[XS]**.
  - 4541 • urn:oasis:names:tc:xacml:3.0:function:dateTime-from-string
 

4542 This function SHALL take one argument of data-type  
 4543 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an  
 4544 "http://www.w3.org/2001/XMLSchema#dateTime". The result SHALL be the string converted to a  
 4545 dateTime. If the argument is not a valid lexical representation of a dateTime, then the result  
 4546 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
  - 4547 urn:oasis:names:tc:xacml:3.0:function:string-from-dateTime
 

4548 This function SHALL take one argument of data-type  
 4549 "http://www.w3.org/2001/XMLSchema#dateTime", and SHALL return an  
 4550 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the dateTime converted to a  
 4551 string in the canonical form specified in **[XS]**.
  - 4552 • urn:oasis:names:tc:xacml:3.0:function:anyURI-from-string
 

4553 This function SHALL take one argument of data-type  
 4554 "http://www.w3.org/2001/XMLSchema#string", and SHALL return a  
 4555 "http://www.w3.org/2001/XMLSchema#anyURI". The result SHALL be the URI constructed by  
 4556 converting the argument to an URI. If the argument is not a valid lexical representation of a URI,  
 4557 then the result SHALL be Indeterminate with status code  
 4558 urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- 4559 • urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI
- 4560     This function SHALL take one argument of data-type
- 4561     "http://www.w3.org/2001/XMLSchema#anyURI", and SHALL return an
- 4562     "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the URI converted to a
- 4563     string in the form it was originally represented in XML form.
  
- 4564 • urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-from-string
- 4565     This function SHALL take one argument of data-type
- 4566     "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
- 4567     "http://www.w3.org/2001/XMLSchema#dayTimeDuration ". The result SHALL be the string
- 4568     converted to a dayTimeDuration. If the argument is not a valid lexical representation of a
- 4569     dayTimeDuration, then the result SHALL be Indeterminate with status code
- 4570     urn:oasis:names:tc:xacml:1.0:status:syntax-error.
  
- 4571 • urn:oasis:names:tc:xacml:3.0:function:string-from-dayTimeDuration
- 4572     This function SHALL take one argument of data-type
- 4573     "http://www.w3.org/2001/XMLSchema#dayTimeDuration ", and SHALL return an
- 4574     "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the dayTimeDuration
- 4575     converted to a string in the canonical form specified in **[XPathFunc]**.
  
- 4576 • urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-from-string
- 4577     This function SHALL take one argument of data-type
- 4578     "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
- 4579     "http://www.w3.org/2001/XMLSchema#yearMonthDuration". The result SHALL be the string
- 4580     converted to a yearMonthDuration. If the argument is not a valid lexical representation of a
- 4581     yearMonthDuration, 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-yearMonthDuration
- 4584     This function SHALL take one argument of data-type
- 4585     "http://www.w3.org/2001/XMLSchema#yearMonthDuration", and SHALL return an
- 4586     "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the yearMonthDuration
- 4587     converted to a string in the canonical form specified in **[XPathFunc]**.
  
- 4588 • urn:oasis:names:tc:xacml:3.0:function:x500Name-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     "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". The result SHALL be the string converted
- 4592     to an x500Name. If the argument is not a valid lexical representation of a X500Name, then the
- 4593     result SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
  
- 4594 • urn:oasis:names:tc:xacml:3.0:function:string-from-x500Name
- 4595     This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:1.0:data-
- 4596     type:x500Name", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result
- 4597     SHALL be the x500Name converted to a string in the form it was originally represented in XML
- 4598     form..
  
- 4599 • urn:oasis:names:tc:xacml:3.0:function:rfc822Name-from-string
- 4600     This function SHALL take one argument of data-type
- 4601     "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
- 4602     "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". The result SHALL be the string converted
- 4603     to an rfc822Name. If the argument is not a valid lexical representation of an rfc822Name, then the
- 4604     result SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
  
- 4605 • urn:oasis:names:tc:xacml:3.0:function:string-from-rfc822Name
- 4606     This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:1.0:data-
- 4607     type:rfc822Name", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The

- 4608 result SHALL be the rfc822Name converted to a string in the form it was originally represented in  
4609 XML form.
- 4610 • urn:oasis:names:tc:xacml:3.0:function:ipAddress-from-string
    - 4611 This function SHALL take one argument of data-type
    - 4612 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
    - 4613 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". The result SHALL be the string converted to
    - 4614 an ipAddress. If the argument is not a valid lexical representation of an ipAddress, then the result
    - 4615 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
  - 4616 • urn:oasis:names:tc:xacml:3.0:function:string-from-ipAddress
    - 4617 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:2.0:data-
    - 4618 type:ipAddress", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result
    - 4619 SHALL be the ipAddress converted to a string in the form it was originally represented in XML
    - 4620 form.
  - 4621 • urn:oasis:names:tc:xacml:3.0:function:dnsName-from-string
    - 4622 This function SHALL take one argument of data-type
    - 4623 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
    - 4624 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". The result SHALL be the string converted to
    - 4625 a dnsName. If the argument is not a valid lexical representation of a dnsName, then the result
    - 4626 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
  - 4627 • urn:oasis:names:tc:xacml:3.0:function:string-from-dnsName
    - 4628 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:2.0:data-
    - 4629 type:dnsName", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result
    - 4630 SHALL be the dnsName converted to a string in the form it was originally represented in XML
    - 4631 form.
  - 4632 • urn:oasis:names:tc:xacml:3.0:function:string-starts-with
    - 4633 This function SHALL take two arguments of data-type
    - 4634 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a
    - 4635 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string
    - 4636 begins with the first string, and false otherwise. Equality testing SHALL be done as defined for
    - 4637 urn:oasis:names:tc:xacml:1.0:function:string-equal.
  - 4638 • urn:oasis:names:tc:xacml:3.0:function:anyURI-starts-with
    - 4639 This function SHALL take a first argument of data-
    - 4640 type"http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type
    - 4641 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a
    - 4642 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted
    - 4643 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI begins with the string,
    - 4644 and false otherwise. Equality testing SHALL be done as defined for
    - 4645 urn:oasis:names:tc:xacml:1.0:function:string-equal.
  - 4646 • urn:oasis:names:tc:xacml:3.0:function:string-ends-with
    - 4647 This function SHALL take two arguments of data-type
    - 4648 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a
    - 4649 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string
    - 4650 ends with the first string, and false otherwise. Equality testing SHALL be done as defined for
    - 4651 urn:oasis:names:tc:xacml:1.0:function:string-equal.
  - 4652 • urn:oasis:names:tc:xacml:3.0:function:anyURI-ends-with
    - 4653 This function SHALL take a first argument of data-type
    - 4654 "http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type
    - 4655 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a
    - 4656 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted
    - 4657 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI ends with the string,



- 4658 and false otherwise. Equality testing SHALL be done as defined for  
 4659 urn:oasis:names:tc:xacml:1.0:function:string-equal.
- 4660 • urn:oasis:names:tc:xacml:3.0:function:string-contains
 

4661 This function SHALL take two arguments of data-type  
 4662 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a  
 4663 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string  
 4664 contains the first string, and false otherwise. Equality testing SHALL be done as defined for  
 4665 urn:oasis:names:tc:xacml:1.0:function:string-equal.
  - 4666 • urn:oasis:names:tc:xacml:3.0:function:anyURI-contains
 

4667 This function SHALL take a first argument of data-type  
 4668 "http://www.w3.org/2001/XMLSchema#string" and a second argument of data-type  
 4669 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a  
 4670 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted  
 4671 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI contains the string, and  
 4672 false otherwise. Equality testing SHALL be done as defined for  
 4673 urn:oasis:names:tc:xacml:1.0:function:string-equal.
  - 4674 • urn:oasis:names:tc:xacml:3.0:function:string-substring
 

4675 This function SHALL take a first argument of data-type  
 4676 "http://www.w3.org/2001/XMLSchema#string" and a second and a third argument of type  
 4677 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return a  
 4678 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the substring of the first  
 4679 argument beginning at the position given by the second argument and ending at the position  
 4680 before the position given by the third argument. The first character of the string has position zero.  
 4681 The negative integer value -1 given for the third arguments indicates the end of the string. If the  
 4682 second or third arguments are out of bounds, then the function MUST evaluate to Indeterminate  
 4683 with a status code of urn:oasis:names:tc:xacml:1.0:status:processing-error.
  - 4684 • urn:oasis:names:tc:xacml:3.0:function:anyURI-substring
 

4685 This function SHALL take a first argument of data-type  
 4686 "http://www.w3.org/2001/XMLSchema#anyURI" and a second and a third argument of type  
 4687 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return a  
 4688 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the substring of the first  
 4689 argument converted to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI  
 4690 beginning at the position given by the second argument and ending at the position before the  
 4691 position given by the third argument. The first character of the URI converted to a string has  
 4692 position zero. The negative integer value -1 given for the third arguments indicates the end of the  
 4693 string. If the second or third arguments are out of bounds, then the function MUST evaluate to  
 4694 Indeterminate with a status code of  
 4695 urn:oasis:names:tc:xacml:1.0:status:processing-error. If the resulting substring  
 4696 is not syntactically a valid URI, then the function MUST evaluate to Indeterminate with a status  
 4697 code of urn:oasis:names:tc:xacml:1.0:status:processing-error.

4698

### 4699 **A.3.10 Bag functions**

4700 These functions operate on a **bag** of 'type' values, where type is one of the primitive data-types, and x.x  
 4701 is a version of XACML where the function has been defined. Some additional conditions defined for  
 4702 each function below SHALL cause the expression to evaluate to "Indeterminate".

- 4703 • urn:oasis:names:tc:xacml:x.x:function:type-one-and-only
 

4704 This function SHALL take a **bag** of 'type' values as an argument and SHALL return a value of  
 4705 'type'. It SHALL return the only value in the **bag**. If the **bag** does not have one and only one  
 4706 value, then the expression SHALL evaluate to "Indeterminate".

- 4707 • urn:oasis:names:tc:xacml:x.x:function:type-bag-size
- 4708     This function SHALL take a **bag** of 'type' values as an argument and SHALL return an
- 4709     "http://www.w3.org/2001/XMLSchema#integer" indicating the number of values in the **bag**.
- 4710 • urn:oasis:names:tc:xacml:x.x:function:type-is-in
- 4711     This function SHALL take an argument of 'type' as the first argument and a **bag** of 'type' values
- 4712     as the second argument and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean".
- 4713     The function SHALL evaluate to "True" if and only if the first argument matches by the
- 4714     "urn:oasis:names:tc:xacml:x.x:function:type-equal" any value in the **bag**. Otherwise, it SHALL
- 4715     return "False".
- 4716 • urn:oasis:names:tc:xacml:x.x:function:type-bag
- 4717     This function SHALL take any number of arguments of 'type' and return a **bag** of 'type' values
- 4718     containing the values of the arguments. An application of this function to zero arguments SHALL
- 4719     produce an empty **bag** of the specified data-type.

### 4720 A.3.11 Set functions

4721 These functions operate on **bags** mimicking sets by eliminating duplicate elements from a **bag**.

- 4722 • urn:oasis:names:tc:xacml:x.x:function:type-intersection
- 4723     This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4724     **bag** of 'type' values such that it contains only elements that are common between the two **bags**,
- 4725     which is determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal". No duplicates, as
- 4726     determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal", SHALL exist in the result.
- 4727 • urn:oasis:names:tc:xacml:x.x:function:type-at-least-one-member-of
- 4728     This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4729     "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL evaluate to "True" if and
- 4730     only if at least one element of the first argument is contained in the second argument as
- 4731     determined by "urn:oasis:names:tc:xacml:x.x:function:type-is-in".
- 4732 • urn:oasis:names:tc:xacml:x.x:function:type-union
- 4733     This function SHALL take two or more arguments that are both a **bag** of 'type' values. The
- 4734     expression SHALL return a **bag** of 'type' such that it contains all elements of all the argument
- 4735     **bags**. No duplicates, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal",
- 4736     SHALL exist in the result.
- 4737 • urn:oasis:names:tc:xacml:x.x:function:type-subset
- 4738     This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4739     "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
- 4740     argument is a subset of the second argument. Each argument SHALL be considered to have had
- 4741     its duplicates removed, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal",
- 4742     before the subset calculation.
- 4743 • urn:oasis:names:tc:xacml:x.x:function:type-set-equals
- 4744     This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4745     "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return the result of applying
- 4746     "urn:oasis:names:tc:xacml:1.0:function:and" to the application of
- 4747     "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the first and second arguments and the
- 4748     application of "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the second and first
- 4749     arguments.

### 4750 A.3.12 Higher-order bag functions

4751 This section describes functions in XACML that perform operations on **bags** such that functions may be

4752 applied to the **bags** in general.

4753 • urn:oasis:names:tc:xacml:3.0:function:any-of

4754 This function applies a Boolean function between specific primitive values and a **bag** of values,  
4755 and SHALL return "True" if and only if the **predicate** is "True" for at least one element of the **bag**.

4756 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL  
4757 be an <Function> element that names a Boolean function that takes n arguments of primitive  
4758 types. Under the remaining n arguments, n-1 parameters SHALL be values of primitive data-  
4759 types and one SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated as  
4760 if the function named in the <Function> argument were applied to the n-1 non-bag arguments  
4761 and each element of the bag argument and the results are combined with  
4762 "urn:oasis:names:tc:xacml:1.0:function:or".

4763 For example, the following expression SHALL return "True":

```
4764 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">  
4765   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>  
4766   <AttributeValue  
4767     DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4768   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4769     <AttributeValue  
4770       DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>  
4771     <AttributeValue  
4772       DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4773     <AttributeValue  
4774       DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>  
4775     <AttributeValue  
4776       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4777   </Apply>  
4778 </Apply>
```

4779 This expression is "True" because the first argument is equal to at least one of the elements of  
4780 the **bag**, according to the function.

4781 • urn:oasis:names:tc:xacml:3.0:function:all-of

4782 This function applies a Boolean function between a specific primitive value and a **bag** of values,  
4783 and returns "True" if and only if the **predicate** is "True" for every element of the **bag**.

4784 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL  
4785 be a <Function> element that names a Boolean function that takes n arguments of primitive  
4786 types. Under the remaining n arguments, n-1 parameters SHALL be values of primitive data-  
4787 types and one SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated as  
4788 if the function named in the <Function> argument were applied to the n-1 non-bag arguments  
4789 and each element of the bag argument and the results are combined with  
4790 "urn:oasis:names:tc:xacml:1.0:function:and".

4791 For example, the following expression SHALL evaluate to "True":

```
4792 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:all-of">  
4793   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4794     greater-than"/>  
4795   <AttributeValue  
4796     DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>  
4797   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4798     <AttributeValue  
4799       DataType="http://www.w3.org/2001/XMLSchema#integer">9</AttributeValue>  
4800     <AttributeValue  
4801       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>  
4802     <AttributeValue  
4803       DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>  
4804     <AttributeValue  
4805       DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>  
4806   </Apply>  
4807 </Apply>
```

4808 This expression is "True" because the first argument (10) is greater than all of the elements of the  
4809 **bag** (9,3,4 and 2).

4810 • urn:oasis:names:tc:xacml:3.0:function:any-of-any

4811 This function applies a Boolean function on each tuple from the cross product on all bags  
4812 arguments, and returns "True" if and only if the **predicate** is "True" for at least one inside-function  
4813 call.

4814 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL  
4815 be an <Function> element that names a Boolean function that takes n arguments. The  
4816 remaining arguments are either primitive data types or bags of primitive types. The expression  
4817 SHALL be evaluated as if the function named in the <Function> argument was applied between  
4818 every tuple of the cross product on all bags and the primitive values, and the results were  
4819 combined using "urn:oasis:names:tc:xacml:1.0:function:or". The semantics are that the result of  
4820 the expression SHALL be "True" if and only if the applied **predicate** is "True" for at least one  
4821 function call on the tuples from the **bags** and primitive values.

4822 For example, the following expression SHALL evaluate to "True":

```
4823 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of-any">  
4824   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>  
4825   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4826     <AttributeValue  
4827     DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4828     <AttributeValue  
4829     DataType="http://www.w3.org/2001/XMLSchema#string">Mary</AttributeValue>  
4830   </Apply>  
4831   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4832     <AttributeValue  
4833     DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>  
4834     <AttributeValue  
4835     DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4836     <AttributeValue  
4837     DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>  
4838     <AttributeValue  
4839     DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4840   </Apply>  
4841 </Apply>
```

4842 This expression is "True" because at least one of the elements of the first **bag**, namely "Ringo", is  
4843 equal to at least one of the elements of the second **bag**.

4844 • urn:oasis:names:tc:xacml:1.0:function:all-of-any

4845 This function applies a Boolean function between the elements of two **bags**. The expression  
4846 SHALL be "True" if and only if the supplied **predicate** is "True" between each element of the first  
4847 **bag** and any element of the second **bag**.

4848 This function SHALL take three arguments. The first argument SHALL be an <Function>  
4849 element that names a Boolean function that takes two arguments of primitive types. The second  
4850 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a  
4851 primitive data-type. The expression SHALL be evaluated as if the  
4852 "urn:oasis:names:tc:xacml:3.0:function:any-of" function had been applied to each value of the first  
4853 **bag** and the whole of the second **bag** using the supplied xacml:Function, and the results were  
4854 then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

4855 For example, the following expression SHALL evaluate to "True":

```
4856 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-any">  
4857   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4858   greater-than"/>  
4859   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4860     <AttributeValue  
4861     DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
```

```

4862         <AttributeValue
4863         DataType="http://www.w3.org/2001/XMLSchema#integer">20</AttributeValue>
4864         </Apply>
4865         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4866         <AttributeValue
4867         DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4868         <AttributeValue
4869         DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4870         <AttributeValue
4871         DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4872         <AttributeValue
4873         DataType="http://www.w3.org/2001/XMLSchema#integer">19</AttributeValue>
4874         </Apply>
4875     </Apply>

```

4876 This expression is "True" because each of the elements of the first **bag** is greater than at least  
4877 one of the elements of the second **bag**.

4878 • urn:oasis:names:tc:xacml:1.0:function:any-of-all

4879 This function applies a Boolean function between the elements of two **bags**. The expression  
4880 SHALL be "True" if and only if the supplied **predicate** is "True" between each element of the  
4881 second **bag** and any element of the first **bag**.

4882 This function SHALL take three arguments. The first argument SHALL be an <Function>  
4883 element that names a Boolean function that takes two arguments of primitive types. The second  
4884 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a  
4885 primitive data-type. The expression SHALL be evaluated as if the  
4886 "urn:oasis:names:tc:xacml:3.0:function:any-of" function had been applied to each value of the  
4887 second **bag** and the whole of the first **bag** using the supplied xacml:Function, and the results  
4888 were then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

4889 For example, the following expression SHALL evaluate to "True":

```

4890 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-all">
4891     <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
4892     greater-than"/>
4893     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4894     <AttributeValue
4895     DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4896     <AttributeValue
4897     DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4898     </Apply>
4899     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4900     <AttributeValue
4901     DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4902     <AttributeValue
4903     DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
4904     <AttributeValue
4905     DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4906     <AttributeValue
4907     DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
4908     </Apply>
4909 </Apply>

```

4910 This expression is "True" because, for all of the values in the second **bag**, there is a value in the  
4911 first **bag** that is greater.

4912 • urn:oasis:names:tc:xacml:1.0:function:all-of-all

4913 This function applies a Boolean function between the elements of two **bags**. The expression  
4914 SHALL be "True" if and only if the supplied **predicate** is "True" between each and every element  
4915 of the first **bag** collectively against all the elements of the second **bag**.

4916 This function SHALL take three arguments. The first argument SHALL be an <Function>  
4917 element that names a Boolean function that takes two arguments of primitive types. The second

4918 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a  
4919 primitive data-type. The expression is evaluated as if the function named in the <Function>  
4920 element were applied between every element of the second argument and every element of the  
4921 third argument and the results were combined using "urn:oasis:names:tc:xacml:1.0:function:and".  
4922 The semantics are that the result of the expression is "True" if and only if the applied **predicate** is  
4923 "True" for all elements of the first **bag** compared to all the elements of the second **bag**.

4924 For example, the following expression SHALL evaluate to "True":

```
4925 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-all">  
4926 <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4927 greater-than"/>  
4928 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4929 <AttributeValue  
4930 DataType="http://www.w3.org/2001/XMLSchema#integer">6</AttributeValue>  
4931 <AttributeValue  
4932 DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>  
4933 </Apply>  
4934 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4935 <AttributeValue  
4936 DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>  
4937 <AttributeValue  
4938 DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>  
4939 <AttributeValue  
4940 DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>  
4941 <AttributeValue  
4942 DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>  
4943 </Apply>  
4944 </Apply>
```

4945 This expression is "True" because all elements of the first **bag**, "5" and "6", are each greater than  
4946 all of the integer values "1", "2", "3", "4" of the second **bag**.

4947 • urn:oasis:names:tc:xacml:3.0:function:map

4948 This function converts a **bag** of values to another **bag** of values.

4949 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL  
4950 be a <Function> element naming a function that takes a n arguments of a primitive data-type  
4951 and returns a value of a primitive data-type Under the remaining n arguments, n-1 parameters  
4952 SHALL be values of primitive data-types and one SHALL be a **bag** of a primitive data-type. The  
4953 expression SHALL be evaluated as if the function named in the <Function> argument were  
4954 applied to the n-1 non-bag arguments and each element of the bag argument and resulting in a  
4955 **bag** of the converted value. The result SHALL be a **bag** of the primitive data-type that is returned  
4956 by the function named in the <xacml:Function> element.

4957 For example, the following expression,

```
4958 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:map">  
4959 <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-  
4960 normalize-to-lower-case">  
4961 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4962 <AttributeValue  
4963 DataType="http://www.w3.org/2001/XMLSchema#string">Hello</AttributeValue>  
4964 <AttributeValue  
4965 DataType="http://www.w3.org/2001/XMLSchema#string">World!</AttributeValue>  
4966 </Apply>  
4967 </Apply>
```

4968 evaluates to a **bag** containing "hello" and "world!".

### 4969 A.3.13 Regular-expression-based functions

4970 These functions operate on various types using regular expressions and evaluate to  
4971 "http://www.w3.org/2001/XMLSchema#boolean".

- 4972 • urn:oasis:names:tc:xacml:1.0:function:string-regexp-match
- 4973 This function decides a regular expression match. It SHALL take two arguments of
- 4974 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
- 4975 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
- 4976 expression and the second argument SHALL be a general string. The function specification
- 4977 SHALL be that of the "xf:matches" function with the arguments reversed [XF] Section 7.6.2.
- 4978 • urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match
- 4979 This function decides a regular expression match. It SHALL take two arguments; the first is of
- 4980 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
- 4981 "http://www.w3.org/2001/XMLSchema#anyURI". It SHALL return an
- 4982 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
- 4983 expression and the second argument SHALL be a URI. The function SHALL convert the second
- 4984 argument to type "http://www.w3.org/2001/XMLSchema#string" with
- 4985 urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI, then apply
- 4986 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 4987 • urn:oasis:names:tc:xacml:2.0:function:ipAddress-regexp-match
- 4988 This function decides a regular expression match. It SHALL take two arguments; the first is of
- 4989 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
- 4990 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". It SHALL return an
- 4991 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
- 4992 expression and the second argument SHALL be an IPv4 or IPv6 address. The function SHALL
- 4993 convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with
- 4994 urn:oasis:names:tc:xacml:3.0:function:string-from-ipAddress, then apply
- 4995 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 4996 • urn:oasis:names:tc:xacml:2.0:function:dnsName-regexp-match
- 4997 This function decides a regular expression match. It SHALL take two arguments; the first is of
- 4998 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
- 4999 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". It SHALL return an
- 5000 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
- 5001 expression and the second argument SHALL be a DNS name. The function SHALL convert the
- 5002 second argument to type "http://www.w3.org/2001/XMLSchema#string" with
- 5003 urn:oasis:names:tc:xacml:3.0:function:string-from-dnsName, then apply
- 5004 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5005 • urn:oasis:names:tc:xacml:2.0:function:rfc822Name-regexp-match
- 5006 This function decides a regular expression match. It SHALL take two arguments; the first is of
- 5007 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
- 5008 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". It SHALL return an
- 5009 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
- 5010 expression and the second argument SHALL be an RFC 822 name. The function SHALL convert
- 5011 the second argument to type "http://www.w3.org/2001/XMLSchema#string" with
- 5012 urn:oasis:names:tc:xacml:3.0:function:string-from-rfc822Name, then apply
- 5013 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5014 • urn:oasis:names:tc:xacml:2.0:function:x500Name-regexp-match
- 5015 This function decides a regular expression match. It SHALL take two arguments; the first is of
- 5016 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
- 5017 "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". It SHALL return an
- 5018 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
- 5019 expression and the second argument SHALL be an X.500 directory name. The function SHALL
- 5020 convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with
- 5021 urn:oasis:names:tc:xacml:3.0:function:string-from-x500Name, then apply
- 5022 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".



### 5023 **A.3.14 Special match functions**

5024 These functions operate on various types and evaluate to  
5025 "http://www.w3.org/2001/XMLSchema#boolean" based on the specified standard matching algorithm.

- 5026 • urn:oasis:names:tc:xacml:1.0:function:x500Name-match

5027 This function shall take two arguments of "urn:oasis:names:tc:xacml:1.0:data-type:x500Name"  
5028 and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and  
5029 only if the first argument matches some terminal sequence of RDNs from the second argument  
5030 when compared using x500Name-equal.

5031 As an example (non-normative), if the first argument is "O=Medico Corp,C=US" and the second  
5032 argument is "cn=John Smith,o=Medico Corp, c=US", then the function will return "True".

- 5033 • urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match

5034 This function SHALL take two arguments, the first is of data-type  
5035 "http://www.w3.org/2001/XMLSchema#string" and the second is of data-type  
5036 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" and SHALL return an  
5037 "http://www.w3.org/2001/XMLSchema#boolean". This function SHALL evaluate to "True" if the  
5038 first argument matches the second argument according to the following specification.

5039 An RFC822 name consists of a local-part followed by "@" followed by a domain-part. The local-  
5040 part is case-sensitive, while the domain-part (which is usually a DNS name) is not case-sensitive.

5041 The second argument contains a complete rfc822Name. The first argument is a complete or  
5042 partial rfc822Name used to select appropriate values in the second argument as follows.

5043 In order to match a particular address in the second argument, the first argument must specify the  
5044 complete mail address to be matched. For example, if the first argument is  
5045 "Anderson@sun.com", this matches a value in the second argument of "Anderson@sun.com"  
5046 and "Anderson@SUN.COM", but not "Anne.Anderson@sun.com", "anderson@sun.com" or  
5047 "Anderson@east.sun.com".

5048 In order to match any address at a particular domain in the second argument, the first argument  
5049 must specify only a domain name (usually a DNS name). For example, if the first argument is  
5050 "sun.com", this matches a value in the second argument of "Anderson@sun.com" or  
5051 "Baxter@SUN.COM", but not "Anderson@east.sun.com".

5052 In order to match any address in a particular domain in the second argument, the first argument  
5053 must specify the desired domain-part with a leading ".". For example, if the first argument is  
5054 ".east.sun.com", this matches a value in the second argument of "Anderson@east.sun.com" and  
5055 "anne.anderson@ISRG.EAST.SUN.COM" but not "Anderson@sun.com".

### 5056 **A.3.15 XPath-based functions**

5057 This section specifies functions that take XPath expressions for arguments. An XPath expression  
5058 evaluates to a node-set, which is a set of XML nodes that match the expression. A node or node-set is  
5059 not in the formal data-type system of XACML. All comparison or other operations on node-sets are  
5060 performed in isolation of the particular function specified. The context nodes and namespace mappings  
5061 of the XPath expressions are defined by the XPath data-type, see section B.3. The following functions  
5062 are defined:

- 5063 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-count

5064 This function SHALL take an "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" as an  
5065 argument and evaluates to an "http://www.w3.org/2001/XMLSchema#integer". The value  
5066 returned from the function SHALL be the count of the nodes within the node-set that match the  
5067 given XPath expression. If the <Content> element of the category to which the XPath  
5068 expression applies to is not present in the request, this function SHALL return a value of zero.

- 5069 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-equal

5070 This function SHALL take two “urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression”  
5071 arguments and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. The function  
5072 SHALL return "True" if any of the XML nodes in the node-set matched by the first argument  
5073 equals any of the XML nodes in the node-set matched by the second argument. Two nodes are  
5074 considered equal if they have the same identity. If the <Content> element of the category to  
5075 which either XPath expression applies to is not present in the request, this function SHALL return  
5076 a value of “False”.

5077 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-match

5078 This function SHALL take two “urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression”  
5079 arguments and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. This function  
5080 SHALL evaluate to "True" if one of the following two conditions is satisfied: (1) Any of the XML  
5081 nodes in the node-set matched by the first argument is equal to any of the XML nodes in the  
5082 node-set matched by the second argument; (2) any node below any of the XML nodes in the  
5083 node-set matched by the first argument is equal to any of the XML nodes in the node-set  
5084 matched by the second argument. Two nodes are considered equal if they have the same  
5085 identity. If the <Content> element of the category to which either XPath expression applies to is  
5086 not present in the request, this function SHALL return a value of “False”.

5087 NOTE: The first **condition** is equivalent to "xpath-node-equal", and guarantees that "xpath-node-equal" is  
5088 a special case of "xpath-node-match".

### 5089 A.3.16 Other functions

5090 • urn:oasis:names:tc:xacml:3.0:function:access-permitted

5091 This function SHALL take an “http://www.w3.org/2001/XMLSchema#anyURI” and an  
5092 “http://www.w3.org/2001/XMLSchema#string” as arguments. The first argument SHALL be  
5093 interpreted as an **attribute** category. The second argument SHALL be interpreted as the XML  
5094 content of an <Attributes> element with *Category* equal to the first argument. The function  
5095 evaluates to an “http://www.w3.org/2001/XMLSchema#boolean”. This function SHALL return  
5096 "True" if and only if the **policy** evaluation described below returns the value of "Permit".

5097 The following evaluation is described as if the **context** is actually instantiated, but it is only  
5098 required that an equivalent result be obtained.

5099 The function SHALL construct a new **context**, by copying all the information from the current  
5100 **context**, omitting any <Attributes> element with *Category* equal to the first argument. The  
5101 second function argument SHALL be added to the **context** as the content of an <Attributes>  
5102 element with *Category* equal to the first argument.

5103 The function SHALL invoke a complete **policy** evaluation using the newly constructed **context**.  
5104 This evaluation SHALL be completely isolated from the evaluation which invoked the function, but  
5105 shall use all current **policies** and combining algorithms, including any per request **policies**.

5106 The **PDP** SHALL detect any loop which may occur if successive evaluations invoke this function  
5107 by counting the number of total invocations of any instance of this function during any single initial  
5108 invocation of the **PDP**. If the total number of invocations exceeds the bound for such invocations,  
5109 the initial invocation of this function evaluates to Indeterminate with a  
5110 “urn:oasis:names:tc:xacml:1.0:status:processing-error” status code. Also, see the security  
5111 considerations in section 9.1.8.

### 5112 A.3.17 Extension functions and primitive types

5113 Functions and primitive types are specified by string identifiers allowing for the introduction of functions in  
5114 addition to those specified by XACML. This approach allows one to extend the XACML module with  
5115 special functions and special primitive data-types.

5116 In order to preserve the integrity of the XACML evaluation strategy, the result of an extension function  
5117 SHALL depend only on the values of its arguments. Global and hidden parameters SHALL NOT affect

5118 the evaluation of an expression. Functions SHALL NOT have side effects, as evaluation order cannot be  
5119 guaranteed in a standard way.

## 5120 **A.4 Functions, data types, attributes and algorithms planned for** 5121 **deprecation**

5122 The following functions, data types and algorithms have been defined by previous versions of XACML  
5123 and newer and better alternatives are defined in XACML 3.0. Their use is discouraged for new use and  
5124 they are candidates for deprecation in future versions of XACML.

5125 The following xpath based functions have been replaced with equivalent functions which use the new  
5126 urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression datatype instead of strings.

- 5127 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-count
- 5128 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-count
- 5129 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal
- 5130 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-equal
- 5131 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-match
- 5132 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-match

5133 The following URI and string concatenation function has been replaced with a string to URI conversion  
5134 function, which allows the use of the general string functions with URI through string conversion.

- 5135 • urn:oasis:names:tc:xacml:2.0:function:uri-string-concatenate
- 5136 • Replaced by urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI

5137 The following identifiers have been replaced with official identifiers defined by W3C.

- 5138 • <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration>
- 5139 • Replaced with <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- 5140 • <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration>
- 5141 • Replaced with <http://www.w3.org/2001/XMLSchema#yearMonthDuration>

5142 The following functions have been replaced with functions which use the updated dayTimeDuration and  
5143 yearMonthDuration data types.

- 5144 • urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal
- 5145 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-equal
- 5146 • urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-equal
- 5147 • Replaced with urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-equal
- 5148 • urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration
- 5149 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-add-dayTimeDuration
- 5150 • urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration
- 5151 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-add-yearMonthDuration
- 5152 • urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration
- 5153 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-dayTimeDuration
- 5154 • urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-yearMonthDuration
- 5155 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-yearMonthDuration
- 5156 • urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration
- 5157 • Replaced with urn:oasis:names:tc:xacml:3.0:function:date-add-yearMonthDuration
- 5158 • urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration
- 5159 • Replaced with urn:oasis:names:tc:xacml:3.0:function:date-subtract-yearMonthDuration

- 5160 The following attribute identifiers have been replaced with new identifiers
- 5161 • urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address
  - 5162 • Replaced with urn:oasis:names:tc:xacml:3.0:subject:authn-locality:ip-
  - 5163 address
  - 5164 • urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name
  - 5165 • Replaced with urn:oasis:names:tc:xacml:3.0:subject:authn-
  - 5166 locality:dns-name
  - 5167
- 5168 The following combining algorithms have been replaced with new variants which allow for better handling
- 5169 of “Indeterminate” results.
- 5170 • urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides
  - 5171 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides
  - 5172 • urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides
  - 5173 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides
  - 5174 • urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides
  - 5175 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides
  - 5176 • urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides
  - 5177 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides
  - 5178 • urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides
  - 5179 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides
  - 5180 • urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides
  - 5181 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-overrides
  - 5182 • urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-overrides
  - 5183 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-overrides
  - 5184 • urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-overrides
  - 5185 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-overrides

---

## 5186 Appendix B. XACML identifiers (normative)

5187 This section defines standard identifiers for commonly used entities.

### 5188 B.1 XACML namespaces

5189 XACML is defined using this identifier.

5190 `urn:oasis:names:tc:xacml:3.0:core:schema`

### 5191 B.2 Attribute categories

5192 The following **attribute** category identifiers MUST be used when an XACML 2.0 or earlier **policy** or  
5193 request is translated into XACML 3.0.

5194 **Attributes** previously placed in the **Resource**, **Action**, and **Environment** sections of a request are  
5195 placed in an **attribute** category with the following identifiers respectively. It is RECOMMENDED that they  
5196 are used to list **attributes of resources**, **actions**, and the **environment** respectively when authoring  
5197 XACML 3.0 **policies** or requests.

5198 `urn:oasis:names:tc:xacml:3.0:attribute-category:resource`

5199 `urn:oasis:names:tc:xacml:3.0:attribute-category:action`

5200 `urn:oasis:names:tc:xacml:3.0:attribute-category:environment`

5201 **Attributes** previously placed in the **Subject** section of a request are placed in an **attribute** category  
5202 which is identical of the **subject** category in XACML 2.0, as defined below. It is RECOMMENDED that  
5203 they are used to list **attributes of subjects** when authoring XACML 3.0 **policies** or requests.

5204 This identifier indicates the system entity that initiated the **access** request. That is, the initial entity in a  
5205 request chain. If **subject** category is not specified in XACML 2.0, this is the default translation value.

5206 `urn:oasis:names:tc:xacml:1.0:subject-category:access-subject`

5207 This identifier indicates the system entity that will receive the results of the request (used when it is  
5208 distinct from the access-**subject**).

5209 `urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject`

5210 This identifier indicates a system entity through which the **access** request was passed.

5211 `urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject`

5212 This identifier indicates a system entity associated with a local or remote codebase that generated the  
5213 request. Corresponding **subject attributes** might include the URL from which it was loaded and/or the  
5214 identity of the code-signer.

5215 `urn:oasis:names:tc:xacml:1.0:subject-category:codebase`

5216 This identifier indicates a system entity associated with the computer that initiated the **access** request.  
5217 An example would be an IPsec identity.

5218 `urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine`

### 5219 B.3 Data-types

5220 The following identifiers indicate data-types that are defined in Section A.2.

5221 `urn:oasis:names:tc:xacml:1.0:data-type:x500Name`.

5222 `urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name`

5223 `urn:oasis:names:tc:xacml:2.0:data-type:ipAddress`

5224 `urn:oasis:names:tc:xacml:2.0:data-type:dnsName`

5225 `urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression`

- 5226 The following data-type identifiers are defined by XML Schema [XS].
- 5227 <http://www.w3.org/2001/XMLSchema#string>
- 5228 <http://www.w3.org/2001/XMLSchema#boolean>
- 5229 <http://www.w3.org/2001/XMLSchema#integer>
- 5230 <http://www.w3.org/2001/XMLSchema#double>
- 5231 <http://www.w3.org/2001/XMLSchema#time>
- 5232 <http://www.w3.org/2001/XMLSchema#date>
- 5233 <http://www.w3.org/2001/XMLSchema#dateTime>
- 5234 <http://www.w3.org/2001/XMLSchema#anyURI>
- 5235 <http://www.w3.org/2001/XMLSchema#hexBinary>
- 5236 <http://www.w3.org/2001/XMLSchema#base64Binary>
- 5237 The following data-type identifiers correspond to the `dayTimeDuration` and `yearMonthDuration` data-types defined in [XF] Sections 10.3.2 and 10.3.1, respectively.
- 5238 <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- 5239 <http://www.w3.org/2001/XMLSchema#yearMonthDuration>

## 5241 B.4 Subject attributes

- 5242 These identifiers indicate **attributes** of a **subject**. When used, it is RECOMMENDED that they appear within an `<Attributes>` element of the request **context** with a **subject** category (see section B.2).
- 5243
- 5244 At most one of each of these **attributes** is associated with each **subject**. Each **attribute** associated with authentication included within a single `<Attributes>` element relates to the same authentication event.
- 5245
- 5246 This identifier indicates the name of the **subject**.
- 5247 `urn:oasis:names:tc:xacml:1.0:subject:subject-id`
- 5248 This identifier indicates the security domain of the subject. It identifies the administrator and **policy** that manages the name-space in which the **subject** id is administered.
- 5249 `urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier`
- 5250
- 5251 This identifier indicates a public key used to confirm the **subject's** identity.
- 5252 `urn:oasis:names:tc:xacml:1.0:subject:key-info`
- 5253 This identifier indicates the time at which the **subject** was authenticated.
- 5254 `urn:oasis:names:tc:xacml:1.0:subject:authentication-time`
- 5255 This identifier indicates the method used to authenticate the **subject**.
- 5256 `urn:oasis:names:tc:xacml:1.0:subject:authentication-method`
- 5257 This identifier indicates the time at which the **subject** initiated the **access** request, according to the **PEP**.
- 5258 `urn:oasis:names:tc:xacml:1.0:subject:request-time`
- 5259 This identifier indicates the time at which the **subject's** current session began, according to the **PEP**.
- 5260 `urn:oasis:names:tc:xacml:1.0:subject:session-start-time`
- 5261 The following identifiers indicate the location where authentication credentials were activated.
- 5262 This identifier indicates that the location is expressed as an IP address.
- 5263 `urn:oasis:names:tc:xacml:3.0:subject:authn-locality:ip-address`
- 5264 The corresponding **attribute** SHALL be of data-type "`urn:oasis:names:tc:xacml:2.0:data-type:ipAddress`".
- 5265 This identifier indicates that the location is expressed as a DNS name.
- 5266 `urn:oasis:names:tc:xacml:3.0:subject:authn-locality:dns-name`
- 5267 The corresponding **attribute** SHALL be of data-type "`urn:oasis:names:tc:xacml:2.0:data-type:dnsName`".



5268 Where a suitable **attribute** is already defined in LDAP [LDAP-1], [LDAP-2], the XACML identifier SHALL  
5269 be formed by adding the **attribute** name to the URI of the LDAP specification. For example, the **attribute**  
5270 name for the userPassword defined in the RFC 2256 SHALL be:  
5271 `http://www.ietf.org/rfc/rfc2256.txt#userPassword`

## 5272 B.5 Resource attributes

5273 These identifiers indicate **attributes** of the **resource**. When used, it is RECOMMENDED they appear  
5274 within the <Attributes> element of the request **context** with Category  
5275 `urn:oasis:names:tc:xacml:3.0:attribute-category:resource`.

5276 This **attribute** identifies the **resource** to which **access** is requested.

5277 `urn:oasis:names:tc:xacml:1.0:resource:resource-id`

5278 This **attribute** identifies the namespace of the top element(s) of the contents of the <Content> element.  
5279 In the case where the **resource** content is supplied in the request **context** and the **resource**  
5280 namespaces are defined in the **resource**, the **PEP** MAY provide this **attribute** in the request to indicate  
5281 the namespaces of the **resource** content. In this case there SHALL be one value of this **attribute** for  
5282 each unique namespace of the top level elements in the <Content> element. The type of the  
5283 corresponding **attribute** SHALL be “`http://www.w3.org/2001/XMLSchema#anyURI`”.

5284 `urn:oasis:names:tc:xacml:2.0:resource:target-namespace`

## 5285 B.6 Action attributes

5286 These identifiers indicate **attributes** of the **action** being requested. When used, it is RECOMMENDED  
5287 they appear within the <Attributes> element of the request **context** with Category  
5288 `urn:oasis:names:tc:xacml:3.0:attribute-category:action`.

5289 This **attribute** identifies the **action** for which **access** is requested.

5290 `urn:oasis:names:tc:xacml:1.0:action:action-id`

5291 Where the **action** is implicit, the value of the action-id **attribute** SHALL be

5292 `urn:oasis:names:tc:xacml:1.0:action:implied-action`

5293 This **attribute** identifies the namespace in which the action-id **attribute** is defined.

5294 `urn:oasis:names:tc:xacml:1.0:action:action-namespace`

## 5295 B.7 Environment attributes

5296 These identifiers indicate **attributes** of the **environment** within which the **decision request** is to be  
5297 evaluated. When used in the **decision request**, it is RECOMMENDED they appear in the  
5298 <Attributes> element of the request **context** with Category `urn:oasis:names:tc:xacml:3.0:attribute-`  
5299 `category:environment`.

5300 This identifier indicates the current time at the **context handler**. In practice it is the time at which the  
5301 request **context** was created. For this reason, if these identifiers appear in multiple places within a  
5302 <Policy> or <PolicySet>, then the same value SHALL be assigned to each occurrence in the  
5303 evaluation procedure, regardless of how much time elapses between the processing of the occurrences.

5304 `urn:oasis:names:tc:xacml:1.0:environment:current-time`

5305 The corresponding **attribute** SHALL be of data-type “`http://www.w3.org/2001/XMLSchema#time`”.

5306 `urn:oasis:names:tc:xacml:1.0:environment:current-date`

5307 The corresponding **attribute** SHALL be of data-type “`http://www.w3.org/2001/XMLSchema#date`”.

5308 `urn:oasis:names:tc:xacml:1.0:environment:current-dateTime`

5309 The corresponding **attribute** SHALL be of data-type “`http://www.w3.org/2001/XMLSchema#dateTime`”.



## 5310 **B.8 Status codes**

5311 The following status code values are defined.

5312 This identifier indicates success.

5313 urn:oasis:names:tc:xacml:1.0:status:ok

5314 This identifier indicates that all the **attributes** necessary to make a **policy decision** were not available (see Section 5.58).

5316 urn:oasis:names:tc:xacml:1.0:status:missing-attribute

5317 This identifier indicates that some **attribute** value contained a syntax error, such as a letter in a numeric field.

5319 urn:oasis:names:tc:xacml:1.0:status:syntax-error

5320 This identifier indicates that an error occurred during **policy** evaluation. An example would be division by zero.

5322 urn:oasis:names:tc:xacml:1.0:status:processing-error

## 5323 **B.9 Combining algorithms**

5324 The deny-overrides **rule-combining algorithm** has the following value for the ruleCombiningAlgId attribute:

5326 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides

5327 The deny-overrides **policy-combining algorithm** has the following value for the

5328 policyCombiningAlgId attribute:

5329 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides

5330 The permit-overrides **rule-combining algorithm** has the following value for the ruleCombiningAlgId attribute:

5332 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides

5333 The permit-overrides **policy-combining algorithm** has the following value for the

5334 policyCombiningAlgId attribute:

5335 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides

5336 The first-applicable **rule-combining algorithm** has the following value for the ruleCombiningAlgId attribute:

5338 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable

5339 The first-applicable **policy-combining algorithm** has the following value for the

5340 policyCombiningAlgId attribute:

5341 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable

5342 The only-one-applicable-policy **policy-combining algorithm** has the following value for the

5343 policyCombiningAlgId attribute:

5344 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable

5345 The ordered-deny-overrides **rule-combining algorithm** has the following value for the

5346 ruleCombiningAlgId attribute:

5347 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides

5348 The ordered-deny-overrides **policy-combining algorithm** has the following value for the

5349 policyCombiningAlgId attribute:

5350 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-overrides

5352 The ordered-permit-overrides **rule-combining algorithm** has the following value for the

5353 ruleCombiningAlgId attribute:

5354 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-  
5355 overrides

5356 The ordered-permit-overrides **policy-combining algorithm** has the following value for the  
5357 policyCombiningAlgId attribute:

5358 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-  
5359 overrides

5360 The deny-unless-permit **rule-combining algorithm** has the following value for the  
5361 policyCombiningAlgId attribute:

5362 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit

5363 The permit-unless-deny **rule-combining algorithm** has the following value for the  
5364 policyCombiningAlgId attribute:

5365 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-unless-deny

5366 The deny-unless-permit **policy-combining algorithm** has the following value for the  
5367 policyCombiningAlgId attribute:

5368 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit

5369 The permit-unless-deny **policy-combining algorithm** has the following value for the  
5370 policyCombiningAlgId attribute:

5371 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-unless-deny

5372 The legacy deny-overrides **rule-combining algorithm** has the following value for the  
5373 ruleCombiningAlgId attribute:

5374 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides

5375 The legacy deny-overrides **policy-combining algorithm** has the following value for the  
5376 policyCombiningAlgId attribute:

5377 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides

5378 The legacy permit-overrides **rule-combining algorithm** has the following value for the  
5379 ruleCombiningAlgId attribute:

5380 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides

5381 The legacy permit-overrides **policy-combining algorithm** has the following value for the  
5382 policyCombiningAlgId attribute:

5383 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides

5384 The legacy ordered-deny-overrides **rule-combining algorithm** has the following value for the  
5385 ruleCombiningAlgId attribute:

5386 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides

5387 The legacy ordered-deny-overrides **policy-combining algorithm** has the following value for the  
5388 policyCombiningAlgId attribute:

5389 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-  
5390 overrides

5391 The legacy ordered-permit-overrides **rule-combining algorithm** has the following value for the  
5392 ruleCombiningAlgId attribute:

5393 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-  
5394 overrides

5395 The legacy ordered-permit-overrides **policy-combining algorithm** has the following value for the  
5396 policyCombiningAlgId attribute:

5397 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-  
5398 overrides

5399

5400

## Appendix C. Combining algorithms (normative)

5401 This section contains a description of the **rule-** and **policy-combining algorithms** specified by XACML.  
5402 Pseudo code is normative, descriptions in English are non-normative.

5403 The legacy **combining algorithms** are defined in previous versions of XACML, and are retained for  
5404 compatibility reasons. It is RECOMMENDED that the new **combining algorithms** are used instead of the  
5405 legacy **combining algorithms** for new use.

5406 Note that in each case an implementation is conformant as long as it produces the same result as is  
5407 specified here, regardless of how and in what order the implementation behaves internally.

### 5408 C.1 Extended Indeterminate values

5409 Some combining algorithms are defined in terms of an extended set of "Indeterminate" values. See  
5410 section 7.10 for the definition of the Extended Indeterminate values. For these algorithms, the **PDP** MUST  
5411 keep track of the extended set of "Indeterminate" values during **rule** and **policy** combining.

5412 The output of a combining algorithm which does not track the extended set of "Indeterminate" values  
5413 MUST be treated as "Indeterminate{DP}" for the value "Indeterminate" by a combining algorithm which  
5414 tracks the extended set of "Indeterminate" values.

5415 A combining algorithm which does not track the extended set of "Indeterminate" values MUST treat the  
5416 output of a combining algorithm which tracks the extended set of "Indeterminate" values as an  
5417 "Indeterminate" for any of the possible values of the extended set of "Indeterminate".

### 5418 C.2 Deny-overrides

5419 This section defines the "Deny-overrides" **rule-combining algorithm** of a **policy** and **policy-combining**  
5420 **algorithm** of a **policy set**.

5421 This **combining algorithm** makes use of the extended "Indeterminate".

5422 The **rule combining algorithm** defined here has the following identifier:

5423 `urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides`

5424 The **policy combining algorithm** defined here has the following identifier:

5425 `urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides`

5426 The following is a non-normative informative description of this **combining algorithm**.

5427       The deny overrides **combining algorithm** is intended for those cases where a deny  
5428       decision should have priority over a permit decision. This algorithm has the following  
5429       behavior.

- 5430       1. If any decision is "Deny", the result is "Deny".
- 5431       2. Otherwise, if any decision is "Indeterminate{DP}", the result is "Indeterminate{DP}".
- 5432       3. Otherwise, if any decision is "Indeterminate{D}" and another decision is "Indeterminate{P}" or  
5433       Permit, the result is "Indeterminate{DP}".
- 5434       4. Otherwise, if any decision is "Indeterminate{D}", the result is "Indeterminate{D}".
- 5435       5. Otherwise, if any decision is "Permit", the result is "Permit".
- 5436       6. Otherwise, if any decision is "Indeterminate{P}", the result is "Indeterminate{P}".
- 5437       7. Otherwise, the result is "NotApplicable".

5438 The following pseudo-code represents the normative specification of this **combining algorithm**. The  
5439 algorithm is presented here in a form where the input to it is an array with children (the **policies**, **policy**  
5440 **sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set of  
5441 obligations or advice provided by this algorithm is not deterministic.

```

5442 Decision denyOverridesCombiningAlgorithm(Node[] children)
5443 {
5444     Boolean atLeastOneErrorD = false;
5445     Boolean atLeastOneErrorP = false;
5446     Boolean atLeastOneErrorDP = false;
5447     Boolean atLeastOnePermit = false;
5448     for( i=0 ; i < lengthOf(children) ; i++ )
5449     {
5450         Decision decision = children[i].evaluate();
5451         if (decision == Deny)
5452         {
5453             return Deny;
5454         }
5455         if (decision == Permit)
5456         {
5457             atLeastOnePermit = true;
5458             continue;
5459         }
5460         if (decision == NotApplicable)
5461         {
5462             continue;
5463         }
5464         if (decision == Indeterminate{D})
5465         {
5466             atLeastOneErrorD = true;
5467             continue;
5468         }
5469         if (decision == Indeterminate{P})
5470         {
5471             atLeastOneErrorP = true;
5472             continue;
5473         }
5474         if (decision == Indeterminate{DP})
5475         {
5476             atLeastOneErrorDP = true;
5477             continue;
5478         }
5479     }
5480     if (atLeastOneErrorDP)
5481     {
5482         return Indeterminate{DP};
5483     }
5484     if (atLeastOneErrorD && (atLeastOneErrorP || atLeastOnePermit))
5485     {
5486         return Indeterminate{DP};
5487     }
5488     if (atLeastOneErrorD)
5489     {
5490         return Indeterminate{D};
5491     }
5492     if (atLeastOnePermit)
5493     {
5494         return Permit;
5495     }
5496     if (atLeastOneErrorP)
5497     {
5498         return Indeterminate{P};
5499     }
5500     return NotApplicable;
5501 }

```

5502 **Obligations** and **advice** shall be combined as described in Section 7.18.

### 5503 C.3 Ordered-deny-overrides

5504 The following specification defines the "Ordered-deny-overrides" **rule-combining algorithm** of a **policy**.

5505 The behavior of this algorithm is identical to that of the "Deny-overrides" **rule-combining**  
5506 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL  
5507 match the order as listed in the **policy**.

5508 The **rule combining algorithm** defined here has the following identifier:

5509 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides

5510 The following specification defines the "Ordered-deny-overrides" **policy-combining algorithm** of a  
5511 **policy set**.

5512 The behavior of this algorithm is identical to that of the "Deny-overrides" **policy-combining**  
5513 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL  
5514 match the order as listed in the **policy set**.

5515 The **policy combining algorithm** defined here has the following identifier:

5516 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-  
5517 overrides

### 5518 C.4 Permit-overrides

5519 This section defines the "Permit-overrides" **rule-combining algorithm** of a **policy** and **policy-combining**  
5520 **algorithm** of a **policy set**.

5521 This **combining algorithm** makes use of the extended "Indeterminate".

5522 The **rule combining algorithm** defined here has the following identifier:

5523 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides

5524 The **policy combining algorithm** defined here has the following identifier:

5525 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides

5526 The following is a non-normative informative description of this combining algorithm.

5527 The permit overrides **combining algorithm** is intended for those cases where a permit  
5528 decision should have priority over a deny decision. This algorithm has the following  
5529 behavior.

- 5530 1. If any decision is "Permit", the result is "Permit".
- 5531 2. Otherwise, if any decision is "Indeterminate{DP}", the result is "Indeterminate{DP}".
- 5532 3. Otherwise, if any decision is "Indeterminate{P}" and another decision is  
5533 "Indeterminate{D} or Deny, the result is "Indeterminate{DP}".
- 5534 4. Otherwise, if any decision is "Indeterminate{P}", the result is "Indeterminate{P}".
- 5535 5. Otherwise, if any decision is "Deny", the result is "Deny".
- 5536 6. Otherwise, if any decision is "Indeterminate{D}", the result is "Indeterminate{D}".
- 5537 7. Otherwise, the result is "NotApplicable".

5538 The following pseudo-code represents the normative specification of this **combining algorithm**. The  
5539 algorithm is presented here in a form where the input to it is an array with all children (the **policies**, **policy**  
5540 **sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set of  
5541 obligations or advice provided by this algorithm is not deterministic.

```
5542 Decision permitOverridesCombiningAlgorithm(Node[] children)  
5543 {  
5544     Boolean atLeastOneErrorD = false;  
5545     Boolean atLeastOneErrorP = false;  
5546     Boolean atLeastOneErrorDP = false;  
5547     Boolean atLeastOneDeny = false;
```

```

5548 for( i=0 ; i < lengthOf(children) ; i++ )
5549 {
5550     Decision decision = children[i].evaluate();
5551     if (decision == Deny)
5552     {
5553         atLeastOneDeny = true;
5554         continue;
5555     }
5556     if (decision == Permit)
5557     {
5558         return Permit;
5559     }
5560     if (decision == NotApplicable)
5561     {
5562         continue;
5563     }
5564     if (decision == Indeterminate{D})
5565     {
5566         atLeastOneErrorD = true;
5567         continue;
5568     }
5569     if (decision == Indeterminate{P})
5570     {
5571         atLeastOneErrorP = true;
5572         continue;
5573     }
5574     if (decision == Indeterminate{DP})
5575     {
5576         atLeastOneErrorDP = true;
5577         continue;
5578     }
5579 }
5580 if (atLeastOneErrorDP)
5581 {
5582     return Indeterminate{DP};
5583 }
5584 if (atLeastOneErrorP && (atLeastOneErrorD || atLeastOneDeny))
5585 {
5586     return Indeterminate{DP};
5587 }
5588 if (atLeastOneErrorP)
5589 {
5590     return Indeterminate{P};
5591 }
5592 if (atLeastOneDeny)
5593 {
5594     return Deny;
5595 }
5596 if (atLeastOneErrorD)
5597 {
5598     return Indeterminate{D};
5599 }
5600 return NotApplicable;
5601 }

```

5602 **Obligations** and **advice** shall be combined as described in Section 7.18.

## 5603 C.5 Ordered-permit-overrides

5604 The following specification defines the "Ordered-permit-overrides" **rule-combining algorithm** of a **policy**.

5605 The behavior of this algorithm is identical to that of the "Permit-overrides" **rule-combining**  
5606 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL  
5607 match the order as listed in the **policy**.

5608 The **rule combining algorithm** defined here has the following identifier:  
5609 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-  
5610 overrides  
5611 The following specification defines the "Ordered-permit-overrides" **policy-combining algorithm** of a  
5612 **policy set**.  
5613 The behavior of this algorithm is identical to that of the "Permit-overrides" **policy-combining**  
5614 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL  
5615 match the order as listed in the **policy set**.  
5616 The **policy combining algorithm** defined here has the following identifier:  
5617 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-  
5618 overrides

## 5619 C.6 Deny-unless-permit

5620 This section defines the "Deny-unless-permit" **rule-combining algorithm** of a **policy** or **policy-**  
5621 **combining algorithm** of a **policy set**.  
5622 The **rule combining algorithm** defined here has the following identifier:  
5623 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit  
5624 The **policy combining algorithm** defined here has the following identifier:  
5625 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit  
5626 The following is a non-normative informative description of this **combining algorithm**.

5627 The "Deny-unless-permit" **combining algorithm** is intended for those cases where a  
5628 permit decision should have priority over a deny decision, and an "Indeterminate" or  
5629 "NotApplicable" must never be the result. It is particularly useful at the top level in a  
5630 **policy** structure to ensure that a **PDP** will always return a definite "Permit" or "Deny"  
5631 result. This algorithm has the following behavior.

- 5632 1. If any decision is "Permit", the result is "Permit".
- 5633 2. Otherwise, the result is "Deny".

5634 The following pseudo-code represents the normative specification of this **combining algorithm**. The  
5635 algorithm is presented here in a form where the input to it is an array with all the children (the **policies**,  
5636 **policy sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set  
5637 of obligations or advice provided by this algorithm is not deterministic.

```
5638 Decision denyUnlessPermitCombiningAlgorithm(Node[] children)
5639 {
5640     for( i=0 ; i < lengthOf(children) ; i++ )
5641     {
5642         if (children[i].evaluate() == Permit)
5643         {
5644             return Permit;
5645         }
5646     }
5647     return Deny;
5648 }
```

5649 **Obligations** and **advice** shall be combined as described in Section 7.18.

## 5650 C.7 Permit-unless-deny

5651 This section defines the "Permit-unless-deny" **rule-combining algorithm** of a **policy** or **policy-**  
5652 **combining algorithm** of a **policy set**.  
5653 The **rule combining algorithm** defined here has the following identifier:  
5654 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-unless-deny



5655 The **policy combining algorithm** defined here has the following identifier:  
5656 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-unless-deny  
5657 The following is a non-normative informative description of this **combining algorithm**.

5658 The "Permit-unless-deny" **combining algorithm** is intended for those cases where a  
5659 deny decision should have priority over a permit decision, and an "Indeterminate" or  
5660 "NotApplicable" must never be the result. It is particularly useful at the top level in a  
5661 **policy** structure to ensure that a **PDP** will always return a definite "Permit" or "Deny"  
5662 result. This algorithm has the following behavior.

- 5663 1. If any decision is "Deny", the result is "Deny".  
5664 2. Otherwise, the result is "Permit".

5665 The following pseudo-code represents the normative specification of this **combining algorithm**. The  
5666 algorithm is presented here in a form where the input to it is an array with all the children (the **policies**,  
5667 **policy sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set  
5668 of obligations or advice provided by this algorithm is not deterministic.

```
5669 Decision permitUnlessDenyCombiningAlgorithm(Node[] children)
5670 {
5671     for( i=0 ; i < lengthOf(children) ; i++ )
5672     {
5673         if (children[i].evaluate() == Deny)
5674         {
5675             return Deny;
5676         }
5677     }
5678     return Permit;
5679 }
```

5680 **Obligations** and **advice** shall be combined as described in Section 7.18.

## 5681 C.8 First-applicable

5682 This section defines the "First-applicable" **rule-combining algorithm** of a **policy** and **policy-combining**  
5683 **algorithm** of a **policy set**.

5684 The **rule combining algorithm** defined here has the following identifier:

5685 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable

5686 The following is a non-normative informative description of the "First-Applicable" **rule-combining**  
5687 **algorithm** of a **policy**.

5688 Each **rule** SHALL be evaluated in the order in which it is listed in the **policy**. For a particular  
5689 **rule**, if the **target** matches and the **condition** evaluates to "True", then the evaluation of the  
5690 **policy** SHALL halt and the corresponding **effect** of the **rule** SHALL be the result of the evaluation  
5691 of the **policy** (i.e. "Permit" or "Deny"). For a particular **rule** selected in the evaluation, if the  
5692 **target** evaluates to "False" or the **condition** evaluates to "False", then the next **rule** in the order  
5693 SHALL be evaluated. If no further **rule** in the order exists, then the **policy** SHALL evaluate to  
5694 "NotApplicable".

5695 If an error occurs while evaluating the **target** or **condition** of a **rule**, then the evaluation SHALL  
5696 halt, and the **policy** shall evaluate to "Indeterminate", with the appropriate error status.

5697 The following pseudo-code represents the normative specification of this **rule-combining algorithm**.

```
5698 Decision firstApplicableEffectRuleCombiningAlgorithm(Rule[] rules)
5699 {
5700     for( i = 0 ; i < lengthOf(rules) ; i++ )
5701     {
5702         Decision decision = evaluate(rules[i]);
5703         if (decision == Deny)
5704         {
```

```

5705         return Deny;
5706     }
5707     if (decision == Permit)
5708     {
5709         return Permit;
5710     }
5711     if (decision == NotApplicable)
5712     {
5713         continue;
5714     }
5715     if (decision == Indeterminate)
5716     {
5717         return Indeterminate;
5718     }
5719 }
5720 return NotApplicable;
5721 }

```

5722 The **policy combining algorithm** defined here has the following identifier:

5723 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable

5724 The following is a non-normative informative description of the “First-applicable” **policy-combining**  
5725 **algorithm of a policy set**.

5726 Each **policy** is evaluated in the order that it appears in the **policy set**. For a particular **policy**, if  
5727 the **target** evaluates to "True" and the **policy** evaluates to a determinate value of "Permit" or  
5728 "Deny", then the evaluation SHALL halt and the **policy set** SHALL evaluate to the **effect** value of  
5729 that **policy**. For a particular **policy**, if the **target** evaluate to "False", or the **policy** evaluates to  
5730 "NotApplicable", then the next **policy** in the order SHALL be evaluated. If no further **policy** exists  
5731 in the order, then the **policy set** SHALL evaluate to "NotApplicable".

5732 If an error were to occur when evaluating the **target**, or when evaluating a specific **policy**, the  
5733 reference to the **policy** is considered invalid, or the **policy** itself evaluates to "Indeterminate",  
5734 then the evaluation of the **policy-combining algorithm** shall halt, and the **policy set** shall  
5735 evaluate to "Indeterminate" with an appropriate error status.

5736 The following pseudo-code represents the normative specification of this policy-combination algorithm.

```

5737 Decision firstApplicableEffectPolicyCombiningAlgorithm(Policy[] policies)
5738 {
5739     for( i = 0 ; i < lengthOf(policies) ; i++ )
5740     {
5741         Decision decision = evaluate(policies[i]);
5742         if(decision == Deny)
5743         {
5744             return Deny;
5745         }
5746         if(decision == Permit)
5747         {
5748             return Permit;
5749         }
5750         if (decision == NotApplicable)
5751         {
5752             continue;
5753         }
5754         if (decision == Indeterminate)
5755         {
5756             return Indeterminate;
5757         }
5758     }
5759     return NotApplicable;
5760 }

```

5761 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

## 5762 C.9 Only-one-applicable

5763 This section defines the "Only-one-applicable" **policy-combining algorithm** of a **policy set**.

5764 The **policy combining algorithm** defined here has the following identifier:

5765 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable

5766 The following is a non-normative informative description of the "Only-one-applicable" **policy-combining algorithm** of a **policy set**.

5768 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".

5772 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**.

5774 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.

5777 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```
5778 Decision onlyOneApplicablePolicyPolicyCombiningAlogrithm(Policy[] policies)
5779 {
5780     Boolean          atLeastOne      = false;
5781     Policy           selectedPolicy = null;
5782     ApplicableResult appResult;
5783
5784     for ( i = 0; i < lengthOf(policies) ; i++ )
5785     {
5786         appResult = isApplicable(policies[I]);
5787
5788         if ( appResult == Indeterminate )
5789         {
5790             return Indeterminate;
5791         }
5792         if( appResult == Applicable )
5793         {
5794             if ( atLeastOne )
5795             {
5796                 return Indeterminate;
5797             }
5798             else
5799             {
5800                 atLeastOne      = true;
5801                 selectedPolicy = policies[i];
5802             }
5803         }
5804         if ( appResult == NotApplicable )
5805         {
5806             continue;
5807         }
5808     }
5809     if ( atLeastOne )
5810     {
5811         return evaluate(selectedPolicy);
5812     }
5813     else
5814     {
5815         return NotApplicable;
5816     }
5817 }
```

5818 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

## 5819 C.10 Legacy Deny-overrides

5820 This section defines the legacy “Deny-overrides” *rule-combining algorithm* of a *policy* and *policy-*  
5821 *combining algorithm* of a *policy set*.

5822

5823 The *rule combining algorithm* defined here has the following identifier:

5824 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides

5825 The following is a non-normative informative description of this combining algorithm.

5826 The “Deny-overrides” rule combining algorithm is intended for those cases where a deny  
5827 decision should have priority over a permit decision. This algorithm has the following  
5828 behavior.

- 5829 1. If any rule evaluates to "Deny", the result is "Deny".
- 5830 2. Otherwise, if any rule having Effect="Deny" evaluates to "Indeterminate", the result is  
5831 "Indeterminate".
- 5832 3. Otherwise, if any rule evaluates to "Permit", the result is "Permit".
- 5833 4. Otherwise, if any rule having Effect="Permit" evaluates to "Indeterminate", the result is  
5834 "Indeterminate".
- 5835 5. Otherwise, the result is "NotApplicable".

5836 The following pseudo-code represents the normative specification of this *rule-combining algorithm*.

```
5837 Decision denyOverridesRuleCombiningAlgorithm(Rule[] rules)
5838 {
5839     Boolean atLeastOneError = false;
5840     Boolean potentialDeny = false;
5841     Boolean atLeastOnePermit = false;
5842     for( i=0 ; i < lengthOf(rules) ; i++ )
5843     {
5844         Decision decision = evaluate(rules[i]);
5845         if (decision == Deny)
5846         {
5847             return Deny;
5848         }
5849         if (decision == Permit)
5850         {
5851             atLeastOnePermit = true;
5852             continue;
5853         }
5854         if (decision == NotApplicable)
5855         {
5856             continue;
5857         }
5858         if (decision == Indeterminate)
5859         {
5860             atLeastOneError = true;
5861
5862             if (effect(rules[i]) == Deny)
5863             {
5864                 potentialDeny = true;
5865             }
5866             continue;
5867         }
5868     }
5869     if (potentialDeny)
5870     {
5871         return Indeterminate;
5872     }
5873     if (atLeastOnePermit)
5874     {
```

```

5875     return Permit;
5876   }
5877   if (atLeastOneError)
5878   {
5879     return Indeterminate;
5880   }
5881   return NotApplicable;
5882 }

```

5883 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

5884 The **policy combining algorithm** defined here has the following identifier:

5885 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides

5886 The following is a non-normative informative description of this combining algorithm.

5887 The "Deny-overrides" policy combining algorithm is intended for those cases where a  
5888 deny decision should have priority over a permit decision. This algorithm has the  
5889 following behavior.

- 5890 1. If any policy evaluates to "Deny", the result is "Deny".
- 5891 2. Otherwise, if any policy evaluates to "Indeterminate", the result is "Deny".
- 5892 3. Otherwise, if any policy evaluates to "Permit", the result is "Permit".
- 5893 4. Otherwise, the result is "NotApplicable".

5894 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```

5895 Decision denyOverridesPolicyCombiningAlgorithm(Policy[] policies)
5896 {
5897   Boolean atLeastOnePermit = false;
5898   for( i=0 ; i < lengthOf(policies) ; i++ )
5899   {
5900     Decision decision = evaluate(policies[i]);
5901     if (decision == Deny)
5902     {
5903       return Deny;
5904     }
5905     if (decision == Permit)
5906     {
5907       atLeastOnePermit = true;
5908       continue;
5909     }
5910     if (decision == NotApplicable)
5911     {
5912       continue;
5913     }
5914     if (decision == Indeterminate)
5915     {
5916       return Deny;
5917     }
5918   }
5919   if (atLeastOnePermit)
5920   {
5921     return Permit;
5922   }
5923   return NotApplicable;
5924 }

```

5925 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

## 5926 C.11 Legacy Ordered-deny-overrides

5927 The following specification defines the legacy "Ordered-deny-overrides" **rule-combining algorithm** of a  
5928 **policy**.

5929 The behavior of this algorithm is identical to that of the “Deny-overrides” **rule-combining**  
5930 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL  
5931 match the order as listed in the **policy**.

5932 The **rule combining algorithm** defined here has the following identifier:

5933 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides

5934 The following specification defines the legacy “Ordered-deny-overrides” **policy-combining algorithm** of  
5935 a **policy set**.

5936 The behavior of this algorithm is identical to that of the “Deny-overrides” **policy-combining**  
5937 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL  
5938 match the order as listed in the **policy set**.

5939 The **rule combining algorithm** defined here has the following identifier:

5940 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-  
5941 overrides

## 5942 C.12 Legacy Permit-overrides

5943 This section defines the legacy “Permit-overrides” **rule-combining algorithm** of a **policy** and **policy-**  
5944 **combining algorithm** of a **policy set**.

5945 The **rule combining algorithm** defined here has the following identifier:

5946 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides

5947 The following is a non-normative informative description of this combining algorithm.

5948 The “Permit-overrides” rule combining algorithm is intended for those cases where a  
5949 permit decision should have priority over a deny decision. This algorithm has the  
5950 following behavior.

- 5951 1. If any rule evaluates to "Permit", the result is "Permit".
- 5952 2. Otherwise, if any rule having Effect="Permit" evaluates to "Indeterminate" the result is  
5953 "Indeterminate".
- 5954 3. Otherwise, if any rule evaluates to "Deny", the result is "Deny".
- 5955 4. Otherwise, if any rule having Effect="Deny" evaluates to "Indeterminate", the result is  
5956 "Indeterminate".
- 5957 5. Otherwise, the result is "NotApplicable".

5958 The following pseudo-code represents the normative specification of this **rule-combining algorithm**.

```
5959 Decision permitOverridesRuleCombiningAlgorithm(Rule[] rules)
5960 {
5961     Boolean atLeastOneError = false;
5962     Boolean potentialPermit = false;
5963     Boolean atLeastOneDeny = false;
5964     for( i=0 ; i < lengthOf(rules) ; i++ )
5965     {
5966         Decision decision = evaluate(rules[i]);
5967         if (decision == Deny)
5968         {
5969             atLeastOneDeny = true;
5970             continue;
5971         }
5972         if (decision == Permit)
5973         {
5974             return Permit;
5975         }
5976         if (decision == NotApplicable)
5977         {
5978             continue;
5979         }
5980     }
5981 }
```

```

5980     if (decision == Indeterminate)
5981     {
5982         atLeastOneError = true;
5983
5984         if (effect(rules[i]) == Permit)
5985         {
5986             potentialPermit = true;
5987         }
5988         continue;
5989     }
5990 }
5991 if (potentialPermit)
5992 {
5993     return Indeterminate;
5994 }
5995 if (atLeastOneDeny)
5996 {
5997     return Deny;
5998 }
5999 if (atLeastOneError)
6000 {
6001     return Indeterminate;
6002 }
6003 return NotApplicable;
6004 }

```

6005 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

6006 The **policy combining algorithm** defined here has the following identifier:

6007 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides

6008 The following is a non-normative informative description of this combining algorithm.

6009 The "Permit-overrides" policy combining algorithm is intended for those cases where a  
6010 permit decision should have priority over a deny decision. This algorithm has the  
6011 following behavior.

- 6012 1. If any policy evaluates to "Permit", the result is "Permit".
- 6013 2. Otherwise, if any policy evaluates to "Deny", the result is "Deny".
- 6014 3. Otherwise, if any policy evaluates to "Indeterminate", the result is "Indeterminate".
- 6015 4. Otherwise, the result is "NotApplicable".

6016 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```

6017 Decision permitOverridesPolicyCombiningAlgorithm(Policy[] policies)
6018 {
6019     Boolean atLeastOneError = false;
6020     Boolean atLeastOneDeny = false;
6021     for( i=0 ; i < lengthOf(policies) ; i++ )
6022     {
6023         Decision decision = evaluate(policies[i]);
6024         if (decision == Deny)
6025         {
6026             atLeastOneDeny = true;
6027             continue;
6028         }
6029         if (decision == Permit)
6030         {
6031             return Permit;
6032         }
6033         if (decision == NotApplicable)
6034         {
6035             continue;
6036         }
6037         if (decision == Indeterminate)

```



```
6038     {
6039         atLeastOneError = true;
6040         continue;
6041     }
6042 }
6043 if (atLeastOneDeny)
6044 {
6045     return Deny;
6046 }
6047 if (atLeastOneError)
6048 {
6049     return Indeterminate;
6050 }
6051 return NotApplicable;
6052 }
```

6053 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

### 6054 C.13 Legacy Ordered-permit-overrides

6055 The following specification defines the legacy "Ordered-permit-overrides" **rule-combining algorithm** of a  
6056 **policy**.

6057 The behavior of this algorithm is identical to that of the "Permit-overrides" **rule-combining**  
6058 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL  
6059 match the order as listed in the **policy**.

6060 The **rule combining algorithm** defined here has the following identifier:

6061 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-  
6062 overrides

6063 The following specification defines the legacy "Ordered-permit-overrides" **policy-combining algorithm** of  
6064 a **policy set**.

6065 The behavior of this algorithm is identical to that of the "Permit-overrides" **policy-combining**  
6066 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL  
6067 match the order as listed in the **policy set**.

6068 The **policy combining algorithm** defined here has the following identifier:

6069 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-  
6070 overrides

6071

---

## 6072 **Appendix D. Acknowledgements**

6073 The following individuals have participated in the creation of this specification and are gratefully  
6074 acknowledged:

6075  
6076 Anil Saldhana  
6077 Anil Tappetla  
6078 Anne Anderson  
6079 Anthony Nadalin  
6080 Bill Parducci  
6081 Craig Forster  
6082 David Chadwick  
6083 David Staggs  
6084 Dilli Arumugam  
6085 Duane DeCouteau  
6086 Erik Rissanen  
6087 Gareth Richards  
6088 Hal Lockhart  
6089 Jan Herrmann  
6090 John Tolbert  
6091 Ludwig Seitz  
6092 Michiharu Kudo  
6093 Naomaru Itoi  
6094 Paul Tyson  
6095 Prateek Mishra  
6096 Rich Levinson  
6097 Ronald Jacobson  
6098 Seth Proctor  
6099 Sridhar Muppidi  
6100 Tim Moses  
6101 Vernon Murdoch

6102

## Appendix E. Revision History

6103

6104

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 XPath expressions 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 conformace 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>

6105