



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

OASIS Standard incorporating Approved Errata

12 July 2017

Specification URIs

This version:

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

Previous 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>

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), Individual
Hal Lockhart (hal.lockhart@oracle.com), Oracle

Editor:

Erik Rissanen (erik@axiomatics.com), Axiomatics AB

Additional artifacts:

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

- *eXtensible Access Control Markup Language (XACML) Version 3.0 Errata 01*. Edited by Richard C. Hill and Hal Lockhart. 12 July 2017. Approved Errata. <http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os.html>.
- *eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01 (redlined)*. Edited by Erik Rissanen. 12 July 2017. OASIS Standard incorporating Approved Errata. <http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os-redlined.html>.
- XML schema – unmodified from OASIS Standard: <http://docs.oasis-open.org/xacml/3.0/errata01/os/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 Approved 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.

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. 12 July 2017. OASIS Standard incorporating Approved Errata. <http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os-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 **Disjunctive sequence**
- 40 A sequence of *predicates* combined using the logical 'OR' operation
- 41 **Effect**
- 42 The intended consequence of a satisfied *rule* (either "Permit" or "Deny")
- 43 **Environment**
- 44 The set of *attributes* that are relevant to an *authorization decision* and are independent of a
45 particular *subject, resource* or *action*
- 46 **Identifier equality**
- 47 The identifier equality operation which is defined in section 7.20.
- 48 **Issuer**
- 49 A set of *attributes* describing the source of a *policy*
- 50 **Named attribute**
- 51 A specific instance of an *attribute*, determined by the *attribute* name and type, the identity of the
52 *attribute* holder (which may be of type: *subject, resource, action* or *environment*) and
53 (optionally) the identity of the issuing authority
- 54 **Obligation**
- 55 An operation specified in a *rule, policy* or *policy set* that should be performed by the *PEP* in
56 conjunction with the enforcement of an *authorization decision*
- 57 **Policy**
- 58 A set of *rules*, an identifier for the *rule-combining algorithm* and (optionally) a set of
59 *obligations* or *advice*. May be a component of a *policy set*
- 60 **Policy administration point (PAP)**
- 61 The system entity that creates a *policy* or *policy set*
- 62 **Policy-combining algorithm**
- 63 The procedure for combining the *decision* and *obligations* from multiple *policies*
- 64 **Policy decision point (PDP)**
- 65 The system entity that evaluates *applicable policy* and renders an *authorization decision*.
66 This term is defined in a joint effort by the IETF Policy Framework Working Group and the
67 Distributed Management Task Force (DMTF)/Common Information Model (CIM) in [RFC3198].
68 This term corresponds to "Access Decision Function" (ADF) in [ISO10181-3].
- 69 **Policy enforcement point (PEP)**
- 70 The system entity that performs *access control*, by making *decision requests* and enforcing
71 *authorization decisions*. This term is defined in a joint effort by the IETF Policy Framework
72 Working Group and the Distributed Management Task Force (DMTF)/Common Information Model
73 (CIM) in [RFC3198]. This term corresponds to "Access Enforcement Function" (AEF) in
74 [ISO10181-3].
- 75 **Policy information point (PIP)**
- 76 The system entity that acts as a source of *attribute* values
- 77 **Policy set**
- 78 A set of *policies*, other *policy sets*, a *policy-combining algorithm* and (optionally) a set of
79 *obligations* or *advice*. May be a component of another *policy set*
- 80 **Predicate**
- 81 A statement about *attributes* whose truth can be evaluated
- 82 **Resource**

83 Data, service or system component

84 Rule

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

87 Rule-combining algorithm

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

89 Subject

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

91 Target

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

175	[RBAC]	David Ferraiolo and Richard Kuhn, <i>Role-Based Access Controls</i> , 15th National Computer Security Conference, 1992.
176		
177	[RFC2119]	S. Bradner, <i>Key words for use in RFCs to Indicate Requirement Levels</i> , http://www.ietf.org/rfc/rfc2119.txt , IETF RFC 2119, March 1997.
178		
179	[RFC2396]	Berners-Lee T, Fielding R, Masinter L, <i>Uniform Resource Identifiers (URI): Generic Syntax</i> . Available at: http://www.ietf.org/rfc/rfc2396.txt
180		
181	[RFC2732]	Hinden R, Carpenter B, Masinter L, <i>Format for Literal IPv6 Addresses in URL's</i> . Available at: http://www.ietf.org/rfc/rfc2732.txt
182		
183	[RFC3198]	IETF RFC 3198: <i>Terminology for Policy-Based Management</i> , November 2001. http://www.ietf.org/rfc/rfc3198.txt
184		
185	[UAX15]	Mark Davis, Martin Dürst, <i>Unicode Standard Annex #15: Unicode Normalization Forms, Unicode 5.1</i> , available from http://unicode.org/reports/tr15/
186		
187	[UTR36]	Davis, Mark, Suignard, Michel, <i>Unicode Technocal Report #36: Unicode Security Considerations</i> . Available at http://www.unicode.org/reports/tr36/
188		
189	[XACMLAdmin]	OASIS Committee Draft 03, <i>XACML v3.0 Administration and Delegation Profile Version 1.0</i> . 11 March 2010. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-cd-03-en.doc
190		
191		
192	[XACMLv1.0]	OASIS Standard, <i>Extensible access control markup language (XACML) Version 1.0</i> . 18 February 2003. http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf
193		
194		
195	[XACMLv1.1]	OASIS Committee Specification, <i>Extensible access control markup language (XACML) Version 1.1</i> . 7 August 2003. http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf
196		
197		
198	[XF]	<i>XQuery 1.0 and XPath 2.0 Functions and Operators</i> , W3C Recommendation 23 January 2007. Available at: http://www.w3.org/TR/2007/REC-xpath-functions-20070123/
199		
200		
201	[XML]	Bray, Tim, et.al. eds, <i>Extensible Markup Language (XML) 1.0 (Fifth Edition)</i> , W3C Recommendation 26 November 2008, available at http://www.w3.org/TR/2008/REC-xml-20081126/
202		
203		
204	[XMLid]	Marsh, Jonathan, et.al. eds, <i>xml:id Version 1.0</i> . W3C Recommendation 9 September 2005. Available at: http://www.w3.org/TR/2005/REC-xml-id-20050909/
205		
206		
207	[XS]	<i>XML Schema, parts 1 and 2</i> . Available at: http://www.w3.org/TR/xmlschema-1/ and http://www.w3.org/TR/xmlschema-2/
208		
209	[XPath]	<i>XML Path Language (XPath), Version 1.0</i> , W3C Recommendation 16 November 1999. Available at: http://www.w3.org/TR/xpath
210		
211	[XPathFunc]	<i>XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition)</i> , W3C Recommendation 14 December 2010. Available at: http://www.w3.org/TR/2010/REC-xpath-functions-20101214/
212		
213		
214	[XSLT]	<i>XSL Transformations (XSLT) Version 1.0</i> , W3C Recommendation 16 November 1999. Available at: http://www.w3.org/TR/xslt
215		

216 1.5 Non-Normative References

217	[CM]	<i>Character model for the World Wide Web 1.0: Normalization</i> , W3C Working Draft, 27 October 2005, http://www.w3.org/TR/2005/WD-charmod-norm-20051027/ , World Wide Web Consortium.
218		
219		
220	[Hinton94]	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	[Sloman94]	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

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

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

- 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

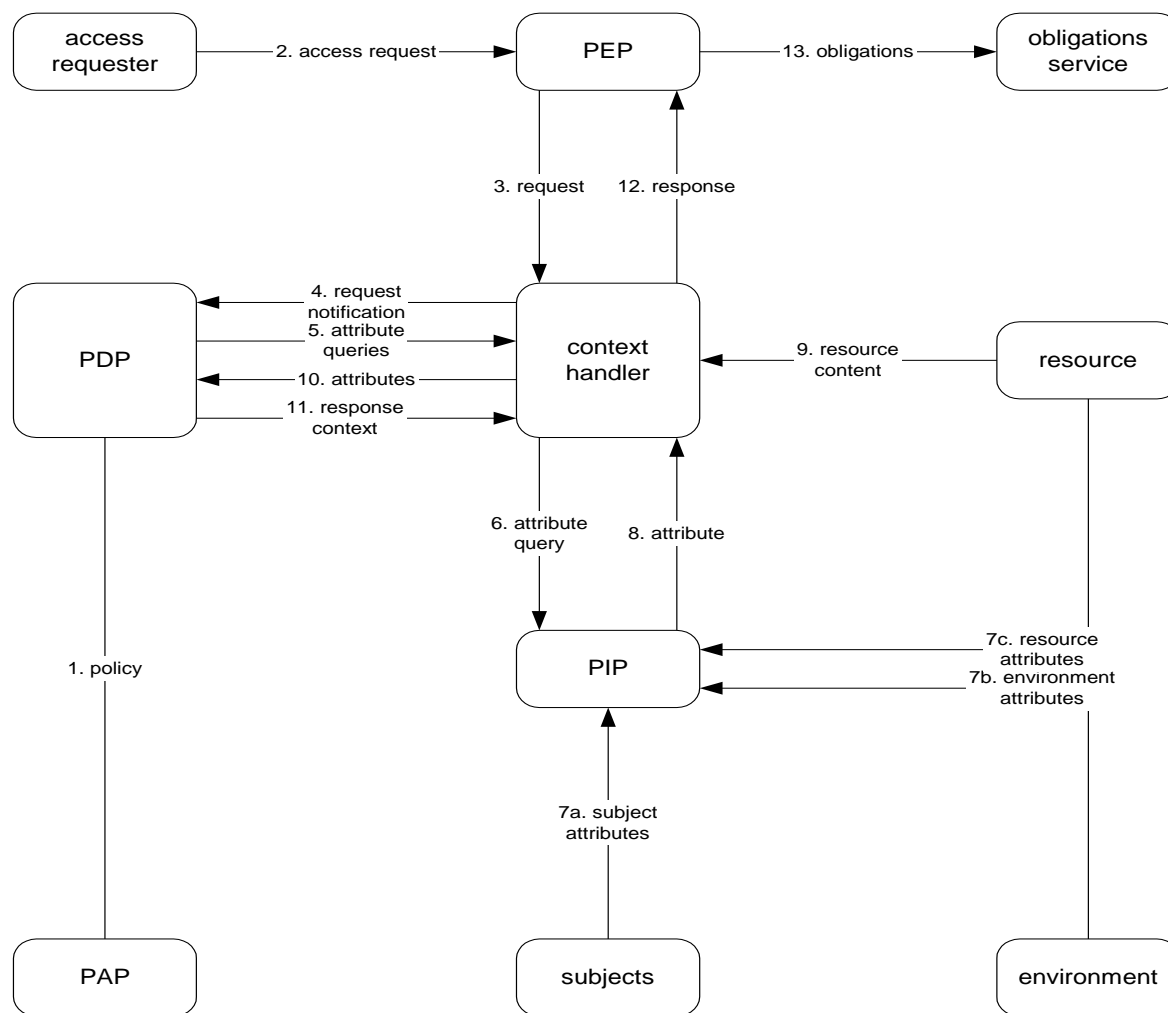
457 In some applications it is helpful to specify supplemental information about a decision. XACML provides
458 facilities to specify supplemental information about a decision with the <Advice> element. Such **advice**
459 may be safely ignored by the **PEP**.

460 3 Models (non-normative)

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

462 3.1 Data-flow model

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



464
465 Figure 1 - Data-flow diagram

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

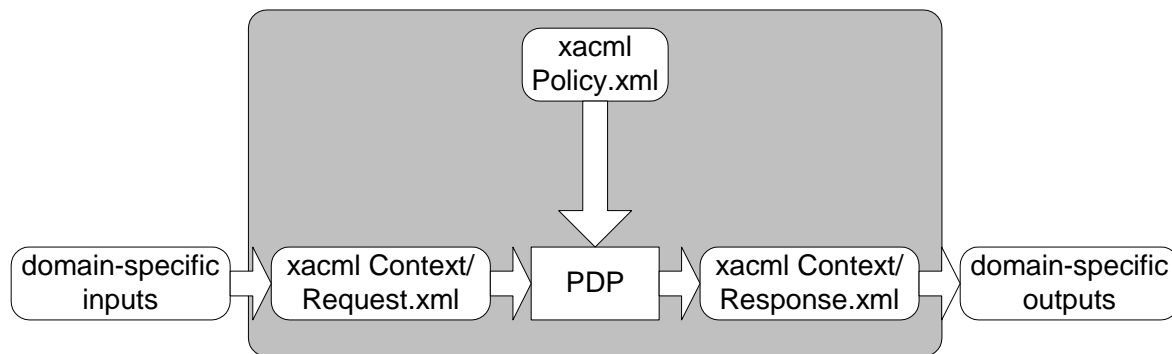
472 The model operates by the following steps.

- 473 1. **PAPs** write **policies** and **policy sets** and make them available to the **PDP**. These **policies** or
474 **policy sets** represent the complete **policy** for a specified **target**.
- 475 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**.

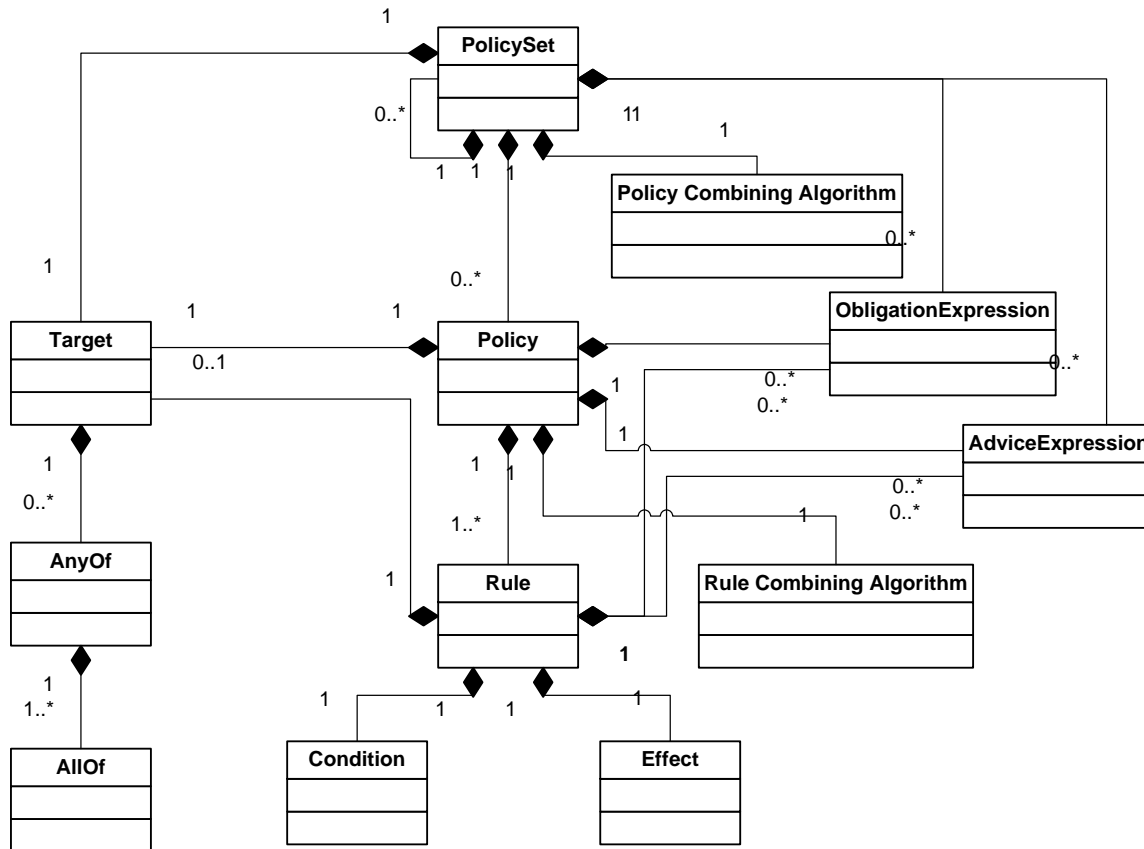
515 **3.3 Policy language model**

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

- 517 • **Rule**;
- 518 • **Policy**; and
- 519 • **Policy set**.

520 These are described in the following sub-sections.

521



522

523 *Figure 3 - Policy language model*

524 **3.3.1 Rule**

525 A **rule** is the most elementary unit of **policy**. It may exist in isolation only within one of the major actors of

526 the XACML domain. In order to exchange **rules** between major actors, they must be encapsulated in a

527 **policy**. A **rule** can be evaluated on the basis of its contents. The main components of a **rule** are:

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

533 These are discussed in the following sub-sections.

534 3.3.1.1 Rule target

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

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

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

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

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

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

560 3.3.1.2 Effect

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

563 3.3.1.3 Condition

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

566 3.3.1.4 Obligation expressions

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

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

571 3.3.1.5 Advice

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

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

576 3.3.2 Policy

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

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

585 3.3.2.1 Policy target

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

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

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

605 3.3.2.2 Rule-combining algorithm

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

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

611 3.3.2.3 Obligation expressions

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

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

616 3.3.2.4 Advice

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

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

621 3.3.3 Policy set

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

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

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

630 3.3.3.1 Policy-combining algorithm

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

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

637 3.3.3.2 Obligation expressions

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

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

643 3.3.3.3 Advice expressions

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

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

649

4 Examples (non-normative)

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

654 4.1 Example one

655 4.1.1 Example policy

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

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

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

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

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

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

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

748 4.1.2 Example request context

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

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

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

```

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

```

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

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

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

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

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

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

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

812 4.1.3 Example response context

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

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

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

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

830 [c6] closes the response *context*.

831 4.2 Example two

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

835 4.2.1 Example medical record instance

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

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

```

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

```

920 4.2.2 Example request context

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

```

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

```

```

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

```

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

1000 [e6] - [e27] **Access subject attributes** are placed in the urn:oasis:names:tc:xacml:1.0:subject-
1001 category:access-subject **attribute** category of the <Request> element. Each **attribute** consists of the
1002 **attribute** meta-data and the **attribute** value. There is only one **subject** involved in this request. This
1003 value of the **attribute** category denotes the identity for which the request was issued.

1004 [e8] - [e13] **Subject** subject-id **attribute**.

1005 [e14] - [e20] **Subject** role **attribute**.

1006 [e21] - [e26] **Subject** physician-id **attribute**.

1007 [e28] - [e56] **Resource attributes** are placed in the urn:oasis:names:tc:xacml:3.0:attribute-
 1008 category:resource **attribute** category of the <Request> element. Each **attribute** consists of **attribute**
 1009 meta-data and an **attribute** value.

1010 [e30] - [e42] **Resource** content. The XML **resource** instance, **access** to all or part of which may be
 1011 requested, is placed here.

1012 [e43] - [e49] The identifier of the **Resource** instance for which **access** is requested, which is an XPath
 1013 expression into the <Content> element that selects the data to be accessed.

1014 [e57] - [e64] **Action attributes** are placed in the urn:oasis:names:tc:xacml:3.0:attribute-category:action
 1015 **attribute** category of the <Request> element.

1016 [e59] - [e63] **Action** identifier.

1017 4.2.3 Example plain-language rules

1018 The following plain-language **rules** are to be enforced:

- 1019 Rule 1: A person, identified by his or her patient number, may read any record for which he or she is
 1020 the designated patient.
- 1021 Rule 2: A person may read any record for which he or she is the designated parent or guardian, and
 1022 for which the patient is under 16 years of age.
- 1023 Rule 3: A physician may write to any medical element for which he or she is the designated primary
 1024 care physician, provided an email is sent to the patient.
- 1025 Rule 4: An administrator shall not be permitted to read or write to medical elements of a patient
 1026 record.

1027 These **rules** may be written by different **PAPs** operating independently, or by a single **PAP**.

1028 4.2.4 Example XACML rule instances

1029 4.2.4.1 Rule 1

1030 **Rule 1** illustrates a simple **rule** with a single <Condition> element. It also illustrates the use of the
 1031 <VariableDefinition> element to define a function that may be used throughout the **policy**. The
 1032 following XACML <Rule> instance expresses **Rule 1**:

```

1033 [f1] <?xml version="1.0" encoding="UTF-8"?>
1034 [f2] <Policy
1035 [f3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1036 [f4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1037 [f5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1038 [f6]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1039 [f7]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:1"
1040 [f8]   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1041     algorithm:deny-overrides"
1042 [f9]   Version="1.0">
1043 [f10]   <PolicyDefaults>
1044 [f11]     <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1045 [f12]   </PolicyDefaults>
1046 [f13]   <Target/>
1047 [f14]   <VariableDefinition VariableId="17590034">
1048 [f15]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1049 [f16]       <Apply
1050 [f17]         FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1051 [f18]           <AttributeDesignator
1052 [f19]             MustBePresent="false"
1053 [f20]             Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
1054             subject"
1055 [f21]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:patient-
1056             number"
1057 [f22]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1058 [f23]           </Apply>
1059 [f24]         </Apply>
1060 [f25]       </Apply>
    
```

```

1061 [f26]         <AttributeSelector
1062 [f27]             MustBePresent="false"
1063 [f28]             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1064 [f29]             Path="md:record/md:patient/md:patient-number/text()"
1065 [f30]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1066 [f31]         </Apply>
1067 [f32]     </Apply>
1068 [f33] </VariableDefinition>
1069 [f34] <Rule
1070 [f35]     RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"
1071 [f36]     Effect="Permit">
1072 [f37]     <Description>
1073 [f38]         A person may read any medical record in the
1074 [f39]         http://www.med.example.com/schemas/record.xsd namespace
1075 [f40]         for which he or she is the designated patient
1076 [f41]     </Description>
1077 [f42]     <Target>
1078 [f43]         <AnyOf>
1079 [f44]             <AllOf>
1080 [f45]                 <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1081 [f46]                     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1082 [f47]                         >urn:example:med:schemas:record</AttributeValue>
1083 [f48]                     <AttributeDesignator
1084 [f49]                         MustBePresent="false"
1085 [f50]                         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1086 [f51]                         AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1087 [f52]                         DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1088 [f53]                 </Match>
1089 [f54]                 <Match
1090 [f55]                     MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1091 [f56]                         <AttributeValue
1092 [f57]                             DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1093 [f58]                             XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1094 [f59]                             >md:record</AttributeValue>
1095 [f60]                         <AttributeDesignator
1096 [f61]                             MustBePresent="false"
1097 [f62]                             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1098 [f63]                             AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1099 [f64]                             DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1100 [f65]                         </Match>
1101 [f66]                     </AllOf>
1102 [f67]                 </AnyOf>
1103 [f68]             <AnyOf>
1104 [f69]                 <AllOf>
1105 [f70]                     <Match
1106 [f71]                         MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1107 [f72]                             <AttributeValue
1108 [f73]                                 DataType="http://www.w3.org/2001/XMLSchema#string"
1109 [f74]                                 >read</AttributeValue>
1110 [f75]                             <AttributeDesignator
1111 [f76]                                 MustBePresent="false"
1112 [f77]                                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1113 [f78]                                 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1114 [f79]                                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1115 [f80]                             </Match>
1116 [f81]                     </AllOf>
1117 [f82]                 </AnyOf>
1118 [f83]             </Target>
1119 [f84]         <Condition>
1120 [f85]             <VariableReference VariableId="17590034"/>
1121 [f86]         </Condition>
1122 [f87]     </Rule>
1123 [f88] </Policy>

```

1124 [f3] - [f6] XML namespace declarations.

1125 [f11] XPath expressions in the **policy** are to be interpreted according to the 1.0 version of the XPath specification.

1127 [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**.

1129 [f15] The `FunctionId` attribute names the function to be used for comparison. In this case, comparison
1130 is done with the “urn:oasis:names:tc:xacml:1.0:function:string-equal” function; this function takes two
1131 arguments of type “http://www.w3.org/2001/XMLSchema#string”.

1132 [f17] The first argument of the variable definition is a function specified by the `FunctionId` attribute.
1133 Since urn:oasis:names:tc:xacml:1.0:function:string-equal takes arguments of type
1134 “http://www.w3.org/2001/XMLSchema#string” and `AttributeDesignator` selects a **bag** of type
1135 “http://www.w3.org/2001/XMLSchema#string”, “urn:oasis:names:tc:xacml:1.0:function:string-one-and-
1136 only” is used. This function guarantees that its argument evaluates to a **bag** containing exactly one
1137 value.

1138 [f18] The `AttributeDesignator` selects a **bag** of values for the patient-number **subject attribute** in
1139 the request **context**.

1140 [f25] The second argument of the variable definition is a function specified by the `FunctionId` attribute.
1141 Since “urn:oasis:names:tc:xacml:1.0:function:string-equal” takes arguments of type
1142 “http://www.w3.org/2001/XMLSchema#string” and the `AttributeSelector` selects a **bag** of type
1143 “http://www.w3.org/2001/XMLSchema#string”, “urn:oasis:names:tc:xacml:1.0:function:string-one-and-
1144 only” is used. This function guarantees that its argument evaluates to a **bag** containing exactly one
1145 value.

1146 [f26] The `<AttributeSelector>` element selects a **bag** of values from the **resource** content using a
1147 free-form XPath expression. In this case, it selects the value of the patient-number in the **resource**.
1148 Note that the namespace prefixes in the XPath expression are resolved with the standard XML
1149 namespace declarations.

1150 [f35] **Rule** identifier.

1151 [f36] **Rule effect** declaration. When a **rule** evaluates to ‘True’ it emits the value of the `Effect` attribute.
1152 This value is then combined with the `Effect` values of other **rules** according to the **rule-combining**
1153 **algorithm**.

1154 [f37] - [f41] Free form description of the **rule**.

1155 [f42] - [f83] A **rule target** defines a set of **decision requests** that the **rule** is intended to evaluate.

1156 [f43] - [f67] The `<AnyOf>` element contains a **disjunctive sequence** of `<AllOf>` elements. In this
1157 example, there is just one.

1158 [f44] - [f66] The `<AllOf>` element encloses the **conjunctive sequence** of `Match` elements. In this
1159 example, there are two.

1160 [f45] - [f53] The first `<Match>` element compares its first and second child elements according to the
1161 matching function. A match is positive if the value of the first argument matches any of the values
1162 selected by the second argument. This match compares the **target** namespace of the requested
1163 document with the value of “urn:example:med:schemas:record”.

1164 [f45] The `MatchId` attribute names the matching function.

1165 [f46] - [f47] Literal **attribute** value to match.

1166 [f48] - [f52] The `<AttributeDesignator>` element selects the **target** namespace from the **resource**
1167 contained in the request **context**. The **attribute** name is specified by the `AttributeId`.

1168 [f54] - [f65] The second `<Match>` element. This match compares the results of two XPath expressions
1169 applied to the `<Content>` element of the **resource** category. The second XPath expression is the
1170 location path to the requested XML element and the first XPath expression is the literal value “md:record”.
1171 The “xpath-node-match” function evaluates to “True” if the requested XML element is below the
1172 “md:record” element.

1173 [f68] - [f82] The `<AnyOf>` element contains a **disjunctive sequence** of `<AllOf>` elements. In this case,
1174 there is just one `<AllOf>` element.

1175 [f69] - [f81] The `<AllOf>` element contains a **conjunctive sequence** of `<Match>` elements. In this case,
1176 there is just one `<Match>` element.

1177 [f70] - [f80] The <Match> element compares its first and second child elements according to the matching
1178 function. The match is positive if the value of the first argument matches any of the values selected by
1179 the second argument. In this case, the value of the action-id **action attribute** in the request **context** is
1180 compared with the literal value "read".

1181 [f84] - [f86] The <Condition> element. A **condition** must evaluate to "True" for the **rule** to be
1182 applicable. This **condition** contains a reference to a variable definition defined elsewhere in the **policy**.

1183 4.2.4.2 Rule 2

1184 **Rule 2** illustrates the use of a mathematical function, i.e. the <Apply> element with functionId
1185 "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration" to calculate the date of the patient's
1186 sixteenth birthday. It also illustrates the use of **predicate** expressions, with the functionId
1187 "urn:oasis:names:tc:xacml:1.0:function:and". This example has one function embedded in the
1188 <Condition> element and another one referenced in a <VariableDefinition> element.

```
1189 [g1] <?xml version="1.0" encoding="UTF-8"?>
1190 [g2] <Policy
1191 [g3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1192 [g4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1193 [g5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1194 [g6]   xmlns:xf="http://www.w3.org/2005/xpath-functions"
1195 [g7]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1196 [g8]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"
1197 [g9]   Version="1.0"
1198 [g10]  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1199      algorithm:deny-overrides">
1200 [g11]  <PolicyDefaults>
1201 [g12]    <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1202 [g13]  </PolicyDefaults>
1203 [g14]  <Target/>
1204 [g15]  <VariableDefinition VariableId="17590035">
1205 [g16]    <Apply
1206 [g17]      FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal">
1207 [g18]      <Apply
1208 [g19]        FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
1209 [g20]          <AttributeDesignator
1210 [g21]            MustBePresent="false"
1211 [g22]            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
1212 [g23]            AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"
1213 [g24]            DataType="http://www.w3.org/2001/XMLSchema#date"/>
1214 [g25]          </Apply>
1215 [g26]        <Apply
1216 [g27]          FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration">
1217 [g28]            <Apply
1218 [g29]              FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
1219 [g30]                <AttributeSelector
1220 [g31]                  MustBePresent="false"
1221 [g32]                  Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1222 [g33]                  Path="md:record/md:patient/md:patientDoB/text()"
1223 [g34]                  DataType="http://www.w3.org/2001/XMLSchema#date"/>
1224 [g35]                </Apply>
1225 [g36]              <AttributeValue
1226 [g37]                DataType="http://www.w3.org/2001/XMLSchema#yearMonthDuration"
1227 [g38]                >P16Y</AttributeValue>
1228 [g39]            </Apply>
1229 [g40]          </Apply>
1230 [g41]        </VariableDefinition>
1231 [g42]      <Rule
1232 [g43]        RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"
1233 [g44]        Effect="Permit">
1234 [g45]        <Description>
1235 [g46]          A person may read any medical record in the
1236 [g47]          http://www.med.example.com/records.xsd namespace
1237 [g48]          for which he or she is the designated parent or guardian,
1238 [g49]          and for which the patient is under 16 years of age
1239 [g50]        </Description>
1240 [g51]        <Target>
1241 [g52]          <AnyOf>
1242 [g53]            <AllOf>
```

```

1243 [g54] <Match
1244 [g55] MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1245 [g56] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1246 [g57] >urn:example:med:schemas:record</AttributeValue>
1247 [g58] <AttributeDesignator
1248 [g59] MustBePresent="false"
1249 [g60] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1250 [g61] AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1251 [g62] DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1252 [g63] </Match>
1253 [g64] <Match
1254 [g65] MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1255 [g66] <AttributeValue
1256 [g67] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1257 [g68] XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1258 [g69] >md:record</AttributeValue>
1259 [g70] <AttributeDesignator
1260 [g71] MustBePresent="false"
1261 [g72] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1262 [g73] AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1263 [g74] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1264 [g75] </Match>
1265 [g76] </AllOf>
1266 [g77] </AnyOf>
1267 [g78] <AnyOf>
1268 [g79] <AllOf>
1269 [g80] <Match
1270 [g81] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1271 [g82] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1272 [g83] >read</AttributeValue>
1273 [g84] <AttributeDesignator
1274 [g85] MustBePresent="false"
1275 [g86] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1276 [g87] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1277 [g88] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1278 [g89] </Match>
1279 [g90] </AllOf>
1280 [g91] </AnyOf>
1281 [g92] </Target>
1282 [g93] <Condition>
1283 [g94] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
1284 [g95] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1285 [g96] <Apply
1286 [g97] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1287 [g98] <AttributeDesignator
1288 [g99] MustBePresent="false"
1289 [g100] Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1290 [g101] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:parent-
guardian-id"
1291 [g102] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1292 [g103] </Apply>
1293 [g104] <Apply
1294 [g105] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1295 [g106] <AttributeSelector
1296 [g107] MustBePresent="false"
1297 [g108] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1298 [g109] Path="md:record/md:parentGuardian/md:parentGuardianId/text()"
1299 [g110] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1300 [g111] </Apply>
1301 [g112] </Apply>
1302 [g113] <VariableReference VariableId="17590035"/>
1303 [g114] </Apply>
1304 [g115] </Condition>
1305 [g116] </Rule>
1306 [g117] </Policy>

```

1308 [g15] - [g41] The <VariableDefinition> element contains part of the **condition** (i.e. is the patient
1309 under 16 years of age?). The patient is under 16 years of age if the current date is less than the date
1310 computed by adding 16 to the patient's date of birth.

1311 [g16] - [g40] "urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal" is used to compare the two date
1312 arguments.

1313 [g18] - [g25] The first date argument uses “urn:oasis:names:tc:xacml:1.0:function:date-one-and-only” to
 1314 ensure that the **bag** of values selected by its argument contains exactly one value of type
 1315 “http://www.w3.org/2001/XMLSchema#date”.

1316 [g20] The current date is evaluated by selecting the “urn:oasis:names:tc:xacml:1.0:environment:current-
 1317 date” **environment attribute**.

1318 [g26] - [g39] The second date argument uses “urn:oasis:names:tc:xacml:1.0:function:date-add-
 1319 yearMonthDuration” to compute the date of the patient’s sixteenth birthday by adding 16 years to the
 1320 patient’s date of birth. The first of its arguments is of type “http://www.w3.org/2001/XMLSchema#date”
 1321 and the second is of type “http://www.w3.org/TR/2007/REC-xpath-functions-20070123/#dt-
 1322 yearMonthDuration”.

1323 [g30] The <AttributeSelector> element selects the patient’s date of birth by taking the XPath
 1324 expression over the **resource** content.

1325 [g36] - [g38] Year Month Duration of 16 years.

1326 [g51] - [g92] **Rule** declaration and **rule target**. See **Rule** 1 in Section 4.2.4.1 for the detailed explanation
 1327 of these elements.

1328 [g93] - [g115] The <Condition> element. The **condition** must evaluate to “True” for the **rule** to be
 1329 applicable. This **condition** evaluates the truth of the statement: the requestor is the designated parent or
 1330 guardian and the patient is under 16 years of age. It contains one embedded <Apply> element and one
 1331 referenced <VariableDefinition> element.

1332 [g94] The **condition** uses the “urn:oasis:names:tc:xacml:1.0:function:and” function. This is a Boolean
 1333 function that takes one or more Boolean arguments (2 in this case) and performs the logical “AND”
 1334 operation to compute the truth value of the expression.

1335 [g95] - [g112] The first part of the **condition** is evaluated (i.e. is the requestor the designated parent or
 1336 guardian?). The function is “urn:oasis:names:tc:xacml:1.0:function:string-equal” and it takes two
 1337 arguments of type “http://www.w3.org/2001/XMLSchema#string”.

1338 [g96] designates the first argument. Since “urn:oasis:names:tc:xacml:1.0:function:string-equal” takes
 1339 arguments of type “http://www.w3.org/2001/XMLSchema#string”,
 1340 “urn:oasis:names:tc:xacml:1.0:function:string-one-and-only” is used to ensure that the **subject attribute**
 1341 “urn:oasis:names:tc:xacml:3.0:example:attribute:parent-guardian-id” in the request **context** contains
 1342 exactly one value.

1343 [g98] designates the first argument. The value of the **subject attribute**
 1344 “urn:oasis:names:tc:xacml:3.0:example:attribute:parent-guardian-id” is selected from the request **context**
 1345 using the <AttributeDesignator> element.

1346 [g104] As above, the “urn:oasis:names:tc:xacml:1.0:function:string-one-and-only” is used to ensure that
 1347 the **bag** of values selected by its argument contains exactly one value of type
 1348 “http://www.w3.org/2001/XMLSchema#string”.

1349 [g106] The second argument selects the value of the <md:parentGuardianId> element from the
 1350 **resource** content using the <AttributeSelector> element. This element contains a free-form XPath
 1351 expression, pointing into the <Content> element of the resource category. Note that all namespace
 1352 prefixes in the XPath expression are resolved with standard namespace declarations. The
 1353 AttributeSelector evaluates to the **bag** of values of type
 1354 “http://www.w3.org/2001/XMLSchema#string”.

1355 [g113] references the <VariableDefinition> element, where the second part of the **condition** is
 1356 defined.

1357 4.2.4.3 Rule 3

1358 **Rule 3** illustrates the use of an **obligation** expression.

```

1359 [h1] <?xml version="1.0" encoding="UTF-8"?>
1360 [h2] <Policy
1361 [h3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1362 [h4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1363 [h5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

```

1364 [h6]      xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
1365 http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
1366 [h7]      xmlns:md="http://www.med.example.com/schemas/record.xsd"
1367 [h8]      PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:3"
1368 [h9]      Version="1.0"
1369 [h10]     RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1370 algorithm:deny-overrides">
1371 [h11]     <Description>
1372 [h12]       Policy for any medical record in the
1373 [h13]       http://www.med.example.com/schemas/record.xsd namespace
1374 [h14]     </Description>
1375 [h15]     <PolicyDefaults>
1376 [h16]       <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1377 [h17]     </PolicyDefaults>
1378 [h18]     <Target>
1379 [h19]       <AnyOf>
1380 [h20]         <AllOf>
1381 [h21]           <Match
1382 [h22]             MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1383 [h23]               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1384 [h24]                 >urn:example:med:schemas:record</AttributeValue>
1385 [h25]               <AttributeDesignator
1386 [h26]                 MustBePresent="false"
1387 [h27]                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1388 [h28]                 AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1389 [h29]                 DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1390 [h30]             </Match>
1391 [h31]           </AllOf>
1392 [h32]         </AnyOf>
1393 [h33]       </Target>
1394 [h34]     <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:3"
1395 [h35]       Effect="Permit">
1396 [h36]       <Description>
1397 [h37]         A physician may write any medical element in a record
1398 [h38]         for which he or she is the designated primary care
1399 [h39]         physician, provided an email is sent to the patient
1400 [h40]       </Description>
1401 [h41]       <Target>
1402 [h42]         <AnyOf>
1403 [h43]           <AllOf>
1404 [h44]             <Match
1405 [h45]               MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1406 [h46]                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1407 [h47]                   >physician</AttributeValue>
1408 [h48]                 <AttributeDesignator
1409 [h49]                   MustBePresent="false"
1410 [h50]                   Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1411 [h51]                   AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
1412 [h52]                   DataType="http://www.w3.org/2001/XMLSchema#string"/>
1413 [h53]                 </Match>
1414 [h54]             </AllOf>
1415 [h55]           </AnyOf>
1416 [h56]         <AnyOf>
1417 [h57]           <AllOf>
1418 [h58]             <Match
1419 [h59]               MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1420 [h60]               <AttributeValue
1421 [h61]                 DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1422 [h62]                 XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1423 [h63]                 >md:record/md:medical</AttributeValue>
1424 [h64]               <AttributeDesignator
1425 [h65]                 MustBePresent="false"
1426 [h66]                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1427 [h67]                 AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1428 [h68]                 DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1429 [h69]               </Match>
1430 [h70]             </AllOf>
1431 [h71]           </AnyOf>
1432 [h72]         <AnyOf>
1433 [h73]           <AllOf>
1434 [h74]             <Match
1435 [h75]               MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1436 [h76]               <AttributeValue

```

```

1437 [h77]         DataType="http://www.w3.org/2001/XMLSchema#string"
1438 [h78]         >write</AttributeValue>
1439 [h79]         <AttributeDesignator
1440 [h80]             MustBePresent="false"
1441 [h81]             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1442 [h82]             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1443 [h83]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1444 [h84]         </Match>
1445 [h85]     </AllOf>
1446 [h86] </AnyOf>
1447 [h87] </Target>
1448 [h88] <Condition>
1449 [h89]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1450 [h90]         <Apply
1451 [h91]             FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1452 [h92]                 <AttributeDesignator
1453 [h93]                     MustBePresent="false"
1454 [h94]                     Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1455 [h95]                     AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:physician-id"
1456 [h96]                     DataType="http://www.w3.org/2001/XMLSchema#string"/>
1457 [h97]                 </Apply>
1458 [h98]             <Apply
1459 [h99]                 FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1460 [h100]                 <AttributeSelector
1461 [h101]                     MustBePresent="false"
1462 [h102]                     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1463 [h103]                     Path="md:record/md:primaryCarePhysician/md:registrationID/text()"
1464 [h104]                     DataType="http://www.w3.org/2001/XMLSchema#string"/>
1465 [h105]                 </Apply>
1466 [h106]             </Apply>
1467 [h107]         </Condition>
1468 [h108] </Rule>
1469 [h109] <ObligationExpressions>
1470 [h110]     <ObligationExpression
1471 [h111]         ObligationId="urn:oasis:names:tc:xacml:example:obligation:email"
1472 [h112]         FulfillOn="Permit">
1473 [h113]         <AttributeAssignmentExpression
1474 [h114]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:mailto">
1475 [h115]             <AttributeSelector
1476 [h116]                 MustBePresent="true"
1477 [h117]                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1478 [h118]                 Path="md:record/md:patient/md:patientContact/md:email"
1479 [h119]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1480 [h120]             </AttributeAssignmentExpression>
1481 [h121]             <AttributeAssignmentExpression
1482 [h122]                 AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
1483 [h123]                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1484 [h124]                     >Your medical record has been accessed by:</AttributeValue>
1485 [h125]                 </AttributeAssignmentExpression>
1486 [h126]             <AttributeAssignmentExpression
1487 [h127]                 AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
1488 [h128]                 <AttributeDesignator
1489 [h129]                     MustBePresent="false"
1490 [h130]                     Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1491 [h131]                     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
1492 [h132]                     DataType="http://www.w3.org/2001/XMLSchema#string"/>
1493 [h133]                 </AttributeAssignmentExpression>
1494 [h134]             </ObligationExpression>
1495 [h135]         </ObligationExpressions>
1496 [h136]     </Policy>

```

1497 [h2] - [h10] The <Policy> element includes standard namespace declarations as well as **policy** specific
1498 parameters, such as PolicyId and RuleCombiningAlgId.

1499 [h8] **Policy** identifier. This parameter allows the **policy** to be referenced by a **policy set**.

1500 [h10] The **Rule-combining algorithm** identifies the algorithm for combining the outcomes of **rule**
1501 evaluation.

1502 [h11] - [h14] Free-form description of the **policy**.

1503 [h18] - [h33] **Policy target**. The **policy target** defines a set of applicable **decision requests**. The
1504 structure of the <Target> element in the <Policy> is identical to the structure of the <Target>

1505 element in the <Rule>. In this case, the **policy target** is the set of all XML **resources** that conform to
 1506 the namespace “urn:example:med:schemas:record”.

1507 [h34] - [h108] The only <Rule> element included in this <Policy>. Two parameters are specified in the
 1508 **rule** header: RuleId and Effect.

1509 [h41] - [h87] The **rule target** further constrains the **policy target**.

1510 [h44] - [h53] The <Match> element targets the **rule** at **subjects** whose
 1511 “urn:oasis:names:tc:xacml:3.0:example:attribute:role” **subject attribute** is equal to “physician”.

1512 [h58] - [h69] The <Match> element targets the **rule** at **resources** that match the XPath expression
 1513 “md:record/md:medical”.

1514 [h74] - [h84] The <Match> element targets the **rule** at **actions** whose
 1515 “urn:oasis:names:tc:xacml:1.0:action:action-id” **action attribute** is equal to “write”.

1516 [h88] - [h107] The <Condition> element. For the **rule** to be applicable to the **decision request**, the
 1517 **condition** must evaluate to “True”. This **condition** compares the value of the
 1518 “urn:oasis:names:tc:xacml:3.0:example:attribute:physician-id” **subject attribute** with the value of the
 1519 <registrationId> element in the medical record that is being accessed.

1520 [h109] - [h134] The <ObligationExpressions> element. **Obligations** are a set of operations that
 1521 must be performed by the **PEP** in conjunction with an **authorization decision**. An **obligation** may be
 1522 associated with a “Permit” or “Deny” **authorization decision**. The element contains a single **obligation**
 1523 expression, which will be evaluated into an obligation when the policy is evaluated.

1524 [h110] - [h133] The <ObligationExpression> element consists of the ObligationId attribute, the
 1525 **authorization decision** value for which it must be fulfilled, and a set of **attribute** assignments.

1526 [h110] The ObligationId attribute identifies the **obligation**. In this case, the **PEP** is required to send
 1527 email.

1528 [h111] The FulfillOn attribute defines the **authorization decision** value for which the **obligation**
 1529 derived from the **obligation** expression must be fulfilled. In this case, the **obligation** must be fulfilled
 1530 when **access** is permitted.

1531 [h112] - [h119] The first parameter indicates where the **PEP** will find the email address in the **resource**.
 1532 The **PDP** will evaluate the <AttributeSelector> and return the result to the **PEP** inside the resulting
 1533 **obligation**.

1534 [h120] - [h123] The second parameter contains literal text for the email body.

1535 [h125] - [h132] The third parameter indicates where the **PEP** will find further text for the email body in the
 1536 **resource**. The **PDP** will evaluate the <AttributeDesignator> and return the result to the **PEP** inside
 1537 the resulting **obligation**.

1538 4.2.4.4 Rule 4

1539 **Rule 4** illustrates the use of the “Deny” **Effect** value, and a <Rule> with no <Condition> element.

```

1540 [i1] <?xml version="1.0" encoding="UTF-8"?>
1541 [i2] <Policy
1542 [i3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1543 [i4]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1544 [i5]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1545 [i6]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:4"
1546 [i7]   Version="1.0"
1547 [i8]   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1548 [i8]   algorithm:deny-overrides">
1549 [i9]   <PolicyDefaults>
1550 [i10]     <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1551 [i11]   </PolicyDefaults>
1552 [i12]   <Target/>
1553 [i13]   <Rule
1554 [i14]     RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:4"
1555 [i15]     Effect="Deny">
1556 [i16]   <Description>
1557 [i17]     An Administrator shall not be permitted to read or write
  
```

```

1558 [i18] medical elements of a patient record in the
1559 [i19] http://www.med.example.com/records.xsd namespace.
1560 [i20] </Description>
1561 [i21] <Target>
1562 [i22] <AnyOf>
1563 [i23] <AllOf>
1564 [i24] <Match
1565 [i25] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1566 [i26] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1567 [i27] >administrator</AttributeValue>
1568 [i28] <AttributeDesignator
1569 [i29] MustBePresent="false"
1570 [i30] Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1571 [i31] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
1572 [i32] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1573 [i33] </Match>
1574 [i34] </AllOf>
1575 [i35] </AnyOf>
1576 [i36] <AnyOf>
1577 [i37] <AllOf>
1578 [i38] <Match
1579 [i39] MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1580 [i40] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1581 [i41] >urn:example:med:schemas:record</AttributeValue>
1582 [i42] <AttributeDesignator
1583 [i43] MustBePresent="false"
1584 [i44] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1585 [i45] AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1586 [i46] DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1587 [i47] </Match>
1588 [i48] <Match
1589 [i49] MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1590 [i50] <AttributeValue
1591 [i51] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1592 [i52] XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1593 [i53] >md:record/md:medical</AttributeValue>
1594 [i54] <AttributeDesignator
1595 [i55] MustBePresent="false"
1596 [i56] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1597 [i57] AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1598 [i58] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1599 [i59] </Match>
1600 [i60] </AllOf>
1601 [i61] </AnyOf>
1602 [i62] <AnyOf>
1603 [i63] <AllOf>
1604 [i64] <Match
1605 [i65] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1606 [i66] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1607 [i67] >read</AttributeValue>
1608 [i68] <AttributeDesignator
1609 [i69] MustBePresent="false"
1610 [i70] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1611 [i71] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1612 [i72] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1613 [i73] </Match>
1614 [i74] </AllOf>
1615 [i75] <AllOf>
1616 [i76] <Match
1617 [i77] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1618 [i78] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1619 [i79] >write</AttributeValue>
1620 [i80] <AttributeDesignator
1621 [i81] MustBePresent="false"
1622 [i82] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1623 [i83] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1624 [i84] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1625 [i85] </Match>
1626 [i86] </AllOf>
1627 [i87] </AnyOf>
1628 [i88] </Target>
1629 [i89] </Rule>
1630 [i90] </Policy>

```


1631 [i13] - [i15] The <Rule> element declaration.

1632 [i15] **Rule Effect**. Every **rule** that evaluates to “True” emits the **rule effect** as its value. This **rule**
 1633 **Effect** is “Deny” meaning that according to this **rule**, **access** must be denied when it evaluates to
 1634 “True”.

1635 [i16] - [i20] Free form description of the **rule**.

1636 [i21] - [i88] **Rule target**. The **Rule target** defines the set of **decision requests** that are applicable to the
 1637 **rule**.

1638 [i24] - [i33] The <Match> element targets the **rule** at **subjects** whose
 1639 “urn:oasis:names:tc:xacml:3.0:example:attribute:role” **subject attribute** is equal to “administrator”.

1640 [i36] - [i61] The <AnyOf> element contains one <AllOf> element, which (in turn) contains two <Match>
 1641 elements. The **target** matches if the **resource** identified by the request **context** matches both **resource**
 1642 match criteria.

1643 [i38] - [i47] The first <Match> element targets the **rule** at **resources** whose
 1644 “urn:oasis:names:tc:xacml:2.0:resource:target-namespace” **resource attribute** is equal to
 1645 “urn:example:med:schemas:record”.

1646 [i48] - [i59] The second <Match> element targets the **rule** at XML elements that match the XPath
 1647 expression “/md:record/md:medical”.

1648 [i62] - [i87] The <AnyOf> element contains two <AllOf> elements, each of which contains one <Match>
 1649 element. The **target** matches if the **action** identified in the request **context** matches either of the **action**
 1650 match criteria.

1651 [i64] - [i85] The <Match> elements **target** the **rule** at **actions** whose
 1652 “urn:oasis:names:tc:xacml:1.0:action:action-id” **action attribute** is equal to “read” or “write”.

1653 This **rule** does not have a <Condition> element.

1654 4.2.4.5 Example PolicySet

1655 This section uses the examples of the previous sections to illustrate the process of combining **policies**.
 1656 The **policy** governing read **access** to medical elements of a record is formed from each of the four **rules**
 1657 described in Section 4.2.3. In plain language, the combined **rule** is:

- 1658 • Either the requestor is the patient; or
- 1659 • the requestor is the parent or guardian and the patient is under 16; or
- 1660 • the requestor is the primary care physician and a notification is sent to the patient; and
- 1661 • the requestor is not an administrator.

1662 The following **policy set** illustrates the combined **policies**. **Policy 3** is included by reference and **policy**
 1663 **2** is explicitly included.

```

1664 [j1]    <?xml version="1.0" encoding="UTF-8"?>
1665 [j2]    <PolicySet
1666 [j3]      xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1667 [j4]      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1668 [j5]      PolicySetId="urn:oasis:names:tc:xacml:3.0:example:policysetid:1"
1669 [j6]      Version="1.0"
1670 [j7]      PolicyCombiningAlgId=
1671 [j8]      "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides">
1672 [j9]    <Description>
1673 [j10]      Example policy set.
1674 [j11]    </Description>
1675 [j12]    <Target>
1676 [j13]      <AnyOf>
1677 [j14]        <AllOf>
1678 [j15]          <Match
1679 [j16]            MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1680 [j17]            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1681 [j18]              >urn:example:med:schema:records</AttributeValue>
1682 [j19]            <AttributeDesignator
1683 [j20]              MustBePresent="false"

```

```

1684 [j21]         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1685 [j22]         AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1686 [j23]         DataType="http://www.w3.org/2001/XMLSchema#string"/>
1687 [j24]         </Match>
1688 [j25]         </AllOf>
1689 [j26]         </AnyOf>
1690 [j27]         </Target>
1691 [j28]         <PolicyIdReference>
1692 [j29]           urn:oasis:names:tc:xacml:3.0:example:policyid:3
1693 [j30]         </PolicyIdReference>
1694 [j31]         <Policy
1695 [j32]           PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"
1696 [j33]           RuleCombiningAlgId=
1697 [j34]             "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides"
1698 [j35]           Version="1.0">
1699 [j36]           <Target/>
1700 [j37]           <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"
1701 [j38]             Effect="Permit">
1702 [j39]           </Rule>
1703 [j40]           <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"
1704 [j41]             Effect="Permit">
1705 [j42]           </Rule>
1706 [j43]           <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:4"
1707 [j44]             Effect="Deny">
1708 [j45]           </Rule>
1709 [j46]         </Policy>
1710 [j47]         </PolicySet>

```

1711 [j2] - [j8] The <PolicySet> element declaration. Standard XML namespace declarations are included.

1712 [j5] The PolicySetId attribute is used for identifying this **policy set** for possible inclusion in another
1713 **policy set**.

1714 [j7] - [j8] The **policy-combining algorithm** identifier. **Policies** and **policy sets** in this **policy set** are
1715 combined according to the specified **policy-combining algorithm** when the **authorization decision** is
1716 computed.

1717 [j9] - [j11] Free form description of the **policy set**.

1718 [j12] - [j27] The **policy set** <Target> element defines the set of **decision requests** that are applicable to
1719 this <PolicySet> element.

1720 [j28] - [j30] PolicyIdReference includes a **policy** by id.

1721 [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>
```

```

1771 <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
1772 <xs:attribute name="Version" type="xacml:VersionType" use="required"/>
1773 <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
1774 <xs:attribute name="MaxDelegationDepth" type="xs:integer" use="optional"/>
1775 </xs:complexType>

```

1776 The <PolicySet> element is of PolicySetType complex type.

1777 The <PolicySet> element contains the following attributes and elements:

1778 PolicySetId [Required]

1779 **Policy set** identifier. It is the responsibility of the **PAP** to ensure that no two **policies** visible to
1780 the **PDP** have the same identifier. This MAY be achieved by following a predefined URN or URI
1781 scheme. If the **policy set** identifier is in the form of a URL, then it MAY be resolvable.

1782 Version [Required]

1783 The version number of the PolicySet.

1784 PolicyCombiningAlgId [Required]

1785 The identifier of the **policy-combining algorithm** by which the <PolicySet>,
1786 <CombinerParameters>, <PolicyCombinerParameters> and
1787 <PolicySetCombinerParameters> components MUST be combined. Standard **policy-**
1788 **combining algorithms** are listed in Appendix Appendix C. Standard **policy-combining**
1789 **algorithm** identifiers are listed in Section B.9.

1790 MaxDelegationDepth [Optional]

1791 If present, limits the depth of delegation which is authorized by this **policy set**. See the delegation
1792 profile [XACMLAdmin].

1793 <Description> [Optional]

1794 A free-form description of the **policy set**.

1795 <PolicyIssuer> [Optional]

1796 **Attributes** of the **issuer** of the **policy set**.

1797 <PolicySetDefaults> [Optional]

1798 A set of default values applicable to the **policy set**. The scope of the <PolicySetDefaults>
1799 element SHALL be the enclosing **policy set**.

1800 <Target> [Required]

1801 The <Target> element defines the applicability of a **policy set** to a set of **decision requests**.

1802 The <Target> element MAY be declared by the creator of the <PolicySet> or it MAY be computed
1803 from the <Target> elements of the referenced <Policy> elements, either as an intersection or
1804 as a union.

1805 <PolicySet> [Any Number]

1806 A **policy set** that is included in this **policy set**.

1807 <Policy> [Any Number]

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

1809 <PolicySetIdReference> [Any Number]

1810 A reference to a **policy set** that MUST be included in this **policy set**. If
1811 <PolicySetIdReference> is a URL, then it MAY be resolvable.

1812 <PolicyIdReference> [Any Number]

1813 A reference to a **policy** that MUST be included in this **policy set**. If the
1814 <PolicyIdReference> is a URL, then it MAY be resolvable.

- 1815 <ObligationExpressions> [Optional]
 1816 Contains the set of <ObligationExpression> elements. See Section 7.18 for a description of
 1817 how the set of **obligations** to be returned by the **PDP** shall be determined.
- 1818 <AdviceExpressions> [Optional]
 1819 Contains the set of <AdviceExpression> elements. See Section 7.18 for a description of how
 1820 the set of **advice** to be returned by the **PDP** shall be determined.
- 1821 <CombinerParameters> [Optional]
 1822 Contains a sequence of <CombinerParameter> elements. The parameters apply to the
 1823 combining algorithm as such and it is up to the specific combining algorithm to interpret them and
 1824 adjust its behavior accordingly.
- 1825 <PolicyCombinerParameters> [Optional]
 1826 Contains a sequence of <CombinerParameter> elements that are associated with a particular
 1827 <Policy> or <PolicyIdReference> element within the <PolicySet>. It is up to the specific
 1828 combining algorithm to interpret them and adjust its behavior accordingly.
- 1829 <PolicySetCombinerParameters> [Optional]
 1830 Contains a sequence of <CombinerParameter> elements that are associated with a particular
 1831 <PolicySet> or <PolicySetIdReference> element within the <PolicySet>. It is up to the
 1832 specific combining algorithm to interpret them and adjust its behavior accordingly.

1833 5.2 Element <Description>

- 1834 The <Description> element contains a free-form description of the <PolicySet>, <Policy>,
 1835 <Rule> or <Apply> element. The <Description> element is of xs:string simple type.

```
1836 <xs:element name="Description" type="xs:string"/>
```

1837 5.3 Element <PolicyIssuer>

- 1838 The <PolicyIssuer> element contains **attributes** describing the issuer of the **policy** or **policy set**.
 1839 The use of the **policy** issuer element is defined in a separate administration profile [**XACMLAdmin**]. A
 1840 PDP which does not implement the administration profile MUST report an error or return an Indeterminate
 1841 result if it encounters this element.

```
1842 <xs:element name="PolicyIssuer" type="xacml:PolicyIssuerType"/>
1843 <xs:complexType name="PolicyIssuerType">
1844   <xs:sequence>
1845     <xs:element ref="xacml:Content" minOccurs="0"/>
1846     <xs:element ref="xacml:Attribute" minOccurs="0" maxOccurs="unbounded"/>
1847   </xs:sequence>
1848 </xs:complexType>
```

- 1849 The <PolicyIssuer> element is of PolicyIssuerType complex type.

- 1850 The <PolicyIssuer> element contains the following elements:

1851 <Content> [Optional]

- 1852 Free form XML describing the issuer. See Section 5.45.

1853 <Attribute> [Zero to many]

- 1854 An **attribute** of the issuer. See Section 5.46.

1855 5.4 Element <PolicySetDefaults>

- 1856 The <PolicySetDefaults> element SHALL specify default values that apply to the <PolicySet>
 1857 element.

```

1858 <xs:element name="PolicySetDefaults" type="xacml:DefaultsType"/>
1859 <xs:complexType name="DefaultsType">
1860   <xs:sequence>
1861     <xs:choice>
1862       <xs:element ref="xacml:XPathVersion">
1863     </xs:choice>
1864   </xs:sequence>
1865 </xs:complexType>

```

1866 <PolicySetDefaults> element is of DefaultsType complex type.

1867 The <PolicySetDefaults> element contains the following elements:

1868 <XPathVersion> [Optional]

1869 Default XPath version.

1870 5.5 Element <XPathVersion>

1871 The <XPathVersion> element SHALL specify the version of the XPath specification to be used by
1872 <AttributeSelector> elements and XPath-based functions in the **policy set** or **policy**.

```

1873 <xs:element name="XPathVersion" type="xs:anyURI"/>

```

1874 The URI for the XPath 1.0 specification is "http://www.w3.org/TR/1999/REC-xpath-19991116".

1875 The URI for the XPath 2.0 specification is "http://www.w3.org/TR/2007/REC-xpath20-20070123".

1876 The <XPathVersion> element is REQUIRED if the XACML enclosing **policy set** or **policy** contains
1877 <AttributeSelector> elements or XPath-based functions.

1878 5.6 Element <Target>

1879 The <Target> element identifies the set of **decision requests** that the parent element is intended to
1880 evaluate. The <Target> element SHALL appear as a child of a <PolicySet> and <Policy> element
1881 and MAY appear as a child of a <Rule> element.

1882 The <Target> element SHALL contain a **conjunctive sequence** of <AnyOf> elements. For the parent
1883 of the <Target> element to be applicable to the **decision request**, there MUST be at least one positive
1884 match between each <AnyOf> element of the <Target> element and the corresponding section of the
1885 <Request> element.

```

1886 <xs:element name="Target" type="xacml:TargetType"/>
1887 <xs:complexType name="TargetType">
1888   <xs:sequence minOccurs="0" maxOccurs="unbounded">
1889     <xs:element ref="xacml:AnyOf"/>
1890   </xs:sequence>
1891 </xs:complexType>

```

1892 The <Target> element is of TargetType complex type.

1893 The <Target> element contains the following elements:

1894 <AnyOf> [Zero to Many]

1895 Matching specification for **attributes** in the **context**. If this element is missing, then the **target**
1896 SHALL match all **contexts**.

1897 5.7 Element <AnyOf>

1898 The <AnyOf> element SHALL contain a **disjunctive sequence** of <AllOf> elements.

```

1899 <xs:element name="AnyOf" type="xacml:AnyOfType"/>
1900 <xs:complexType name="AnyOfType">
1901   <xs:sequence minOccurs="1" maxOccurs="unbounded">
1902     <xs:element ref="xacml:AllOf"/>

```

1903 `</xs:sequence>`
1904 `</xs:complexType>`

1905 The `<AnyOf>` element is of `AnyOfType` complex type.

1906 The `<AnyOf>` element contains the following elements:

1907 `<AllOf>` [One to Many, Required]

1908 See Section 5.8.

1909 5.8 Element `<AllOf>`

1910 The `<AllOf>` element SHALL contain a **conjunctive sequence** of `<Match>` elements.

```
1911 <xs:element name="AllOf" type="xacml:AllOfType"/>
1912 <xs:complexType name="AllOfType">
1913   <xs:sequence minOccurs="1" maxOccurs="unbounded">
1914     <xs:element ref="xacml:Match"/>
1915   </xs:sequence>
1916 </xs:complexType>
```

1917 The `<AllOf>` element is of `AllOfType` complex type.

1918 The `<AllOf>` element contains the following elements:

1919 `<Match>` [One to Many]

1920 A **conjunctive sequence** of individual matches of the **attributes** in the request **context** and the
1921 embedded **attribute** values. See Section 5.9.

1922 5.9 Element `<Match>`

1923 The `<Match>` element SHALL identify a set of entities by matching **attribute** values in an

1924 `<Attributes>` element of the request **context** with the embedded **attribute** value.

```
1925 <xs:element name="Match" type="xacml:MatchType"/>
1926 <xs:complexType name="MatchType">
1927   <xs:sequence>
1928     <xs:element ref="xacml:AttributeValue"/>
1929     <xs:choice>
1930       <xs:element ref="xacml:AttributeDesignator"/>
1931       <xs:element ref="xacml:AttributeSelector"/>
1932     </xs:choice>
1933   </xs:sequence>
1934   <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
1935 </xs:complexType>
```

1936 The `<Match>` element is of `MatchType` complex type.

1937 The `<Match>` element contains the following attributes and elements:

1938 `MatchId` [Required]

1939 Specifies a matching function. The value of this attribute MUST be of type `xs:anyURI` with legal
1940 values documented in Section 7.6.

1941 `<AttributeValue>` [Required]

1942 Embedded **attribute** value.

1943 `<AttributeDesignator>` [Required choice]

1944 MAY be used to identify one or more **attribute** values in an `<Attributes>` element of the
1945 request **context**.

1946 `<AttributeSelector>` [Required choice]

1947 MAY be used to identify one or more **attribute** values in a <Content> element of the request
1948 **context**.

1949 5.10 Element <PolicySetIdReference>

1950 The <PolicySetIdReference> element SHALL be used to reference a <PolicySet> element by id.
1951 If <PolicySetIdReference> is a URL, then it MAY be resolvable to the <PolicySet> element.
1952 However, the mechanism for resolving a **policy set** reference to the corresponding **policy set** is outside
1953 the scope of this specification.

```
1954 <xs:element name="PolicySetIdReference" type="xacml:IdReferenceType"/>  
1955 <xs:complexType name="IdReferenceType">  
1956   <xs:simpleContent>  
1957     <xs:extension base="xs:anyURI">  
1958       <xs:attribute name="xacml:Version"  
1959         type="xacml:VersionMatchType" use="optional"/>  
1960       <xs:attribute name="xacml:EarliestVersion"  
1961         type="xacml:VersionMatchType" use="optional"/>  
1962       <xs:attribute name="xacml:LatestVersion"  
1963         type="xacml:VersionMatchType" use="optional"/>  
1964     </xs:extension>  
1965   </xs:simpleContent>  
1966 </xs:complexType>
```

1967 Element <PolicySetIdReference> is of xacml:IdReferenceType complex type.

1968 IdReferenceType extends the xs:anyURI type with the following attributes:

1969 Version [Optional]

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

1971 EarliestVersion [Optional]

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

1973 LatestVersion [Optional]

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

1975 The matching operation is defined in Section 5.13. Any combination of these attributes MAY be present
1976 in a <PolicySetIdReference>. The referenced **policy set** MUST match all expressions. If none of
1977 these attributes is present, then any version of the **policy set** is acceptable. In the case that more than
1978 one matching version can be obtained, then the most recent one SHOULD be used.

1979 5.11 Element <PolicyIdReference>

1980 The <PolicyIdReference> element SHALL be used to reference a <Policy> element by id. If
1981 <PolicyIdReference> is a URL, then it MAY be resolvable to the <Policy> element. However, the
1982 mechanism for resolving a **policy** reference to the corresponding **policy** is outside the scope of this
1983 specification.

```
1984 <xs:element name="PolicyIdReference" type="xacml:IdReferenceType"/>
```

1985 Element <PolicyIdReference> is of xacml:IdReferenceType complex type (see Section 5.10) .

1986 5.12 Simple type VersionType

1987 Elements of this type SHALL contain the version number of the **policy** or **policy set**.

```
1988 <xs:simpleType name="VersionType">  
1989   <xs:restriction base="xs:string">  
1990     <xs:pattern value="(\d+\.)*\d+"/>  
1991   </xs:restriction>  
1992 </xs:simpleType>
```


1993 The version number is expressed as a sequence of decimal numbers, each separated by a period (.).
1994 'd+' represents a sequence of one or more decimal digits.

1995 5.13 Simple type VersionMatchType

1996 Elements of this type SHALL contain a restricted regular expression matching a version number (see
1997 Section 5.12). The expression SHALL match versions of a referenced **policy** or **policy set** that are
1998 acceptable for inclusion in the referencing **policy** or **policy set**.

```
1999 <xs:simpleType name="VersionMatchType">  
2000   <xs:restriction base="xs:string">  
2001     <xs:pattern value="((\d+|\*)\.)*(\d+|\*|\+)" />  
2002   </xs:restriction>  
2003 </xs:simpleType>
```

2004 A version match is '.'-separated, like a version string. A number represents a direct numeric match. A '*'
2005 means that any single number is valid. A '+' means that any number, and any subsequent numbers, are
2006 valid. In this manner, the following four patterns would all match the version string '1.2.3': '1.2.3', '1.*.3',
2007 '1.2.*' and '1.+'

2008 5.14 Element <Policy>

2009 The <Policy> element is the smallest entity that SHALL be presented to the **PDP** for evaluation.

2010 A <Policy> element may be evaluated, in which case the evaluation procedure defined in Section 7.12
2011 SHALL be used.

2012 The main components of this element are the <Target>, <Rule>, <CombinerParameters>,
2013 <RuleCombinerParameters>, <ObligationExpressions> and <AdviceExpressions>
2014 elements and the RuleCombiningAlgId attribute.

2015 A <Policy> element MAY contain a <PolicyIssuer> element. The interpretation of the
2016 <PolicyIssuer> element is explained in the separate administrative **policy** profile [XACMLAdmin].

2017 The <Target> element defines the applicability of the <Policy> element to a set of **decision requests**.
2018 If the <Target> element within the <Policy> element matches the request **context**, then the
2019 <Policy> element MAY be used by the **PDP** in making its **authorization decision**. See Section 7.12.

2020 The <Policy> element includes a sequence of choices between <VariableDefinition> and
2021 <Rule> elements.

2022 **Rules** included in the <Policy> element MUST be combined by the algorithm specified by the
2023 RuleCombiningAlgId attribute.

2024 The <ObligationExpressions> element contains a set of **obligation** expressions that MUST be
2025 evaluated into **obligations** by the **PDP** and the resulting **obligations** MUST be fulfilled by the **PEP** in
2026 conjunction with the **authorization decision**. If the **PEP** does not understand, or cannot fulfill, any of the
2027 **obligations**, then it MUST act according to the PEP bias. See Section 7.2 and 7.18.

2028 The <AdviceExpressions> element contains a set of **advice** expressions that MUST be evaluated into
2029 **advice** by the **PDP**. The resulting **advice** MAY be safely ignored by the **PEP** in conjunction with the
2030 **authorization decision**. See Section 7.18.

```
2031 <xs:element name="Policy" type="xacml:PolicyType"/>  
2032 <xs:complexType name="PolicyType">  
2033   <xs:sequence>  
2034     <xs:element ref="xacml:Description" minOccurs="0"/>  
2035     <xs:element ref="xacml:PolicyIssuer" minOccurs="0"/>  
2036     <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>  
2037     <xs:element ref="xacml:Target"/>  
2038     <xs:choice maxOccurs="unbounded">  
2039       <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>  
2040       <xs:element ref="xacml:RuleCombinerParameters" minOccurs="0"/>  
2041       <xs:element ref="xacml:VariableDefinition"/>
```

```

2042         <xs:element ref="xacml:Rule"/>
2043     </xs:choice>
2044     <xs:element ref="xacml:ObligationExpressions" minOccurs="0"/>
2045     <xs:element ref="xacml:AdviceExpressions" minOccurs="0"/>
2046 </xs:sequence>
2047 <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
2048 <xs:attribute name="Version" type="xacml:VersionType" use="required"/>
2049 <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
2050 <xs:attribute name="MaxDelegationDepth" type="xs:integer" use="optional"/>
2051 </xs:complexType>

```

2052 The <Policy> element is of PolicyType complex type.

2053 The <Policy> element contains the following attributes and elements:

2054 PolicyId [Required]

2055 **Policy** identifier. It is the responsibility of the **PAP** to ensure that no two **policies** visible to the
2056 **PDP** have the same identifier. This MAY be achieved by following a predefined URN or URI
2057 scheme. If the **policy** identifier is in the form of a URL, then it MAY be resolvable.

2058 Version [Required]

2059 The version number of the **Policy**.

2060 RuleCombiningAlgId [Required]

2061 The identifier of the **rule-combining algorithm** by which the <Policy>,
2062 <CombinerParameters> and <RuleCombinerParameters> components MUST be
2063 combined. Standard **rule-combining algorithms** are listed in Appendix Appendix C. Standard
2064 **rule-combining algorithm** identifiers are listed in Section B.9.

2065 MaxDelegationDepth [Optional]

2066 If present, limits the depth of delegation which is authorized by this **policy**. See the delegation
2067 profile [XACMLAdmin].

2068 <Description> [Optional]

2069 A free-form description of the **policy**. See Section 5.2.

2070 <PolicyIssuer> [Optional]

2071 **Attributes** of the **issuer** of the **policy**.

2072 <PolicyDefaults> [Optional]

2073 Defines a set of default values applicable to the **policy**. The scope of the <PolicyDefaults>
2074 element SHALL be the enclosing **policy**.

2075 <CombinerParameters> [Optional]

2076 A sequence of parameters to be used by the **rule-combining algorithm**. The parameters apply
2077 to the combining algorithm as such and it is up to the specific combining algorithm to interpret
2078 them and adjust its behavior accordingly.

2079 <RuleCombinerParameters> [Optional]

2080 A sequence of <RuleCombinerParameter> elements that are associated with a particular
2081 <Rule> element within the <Policy>.. It is up to the specific combining algorithm to interpret
2082 them and adjust its behavior accordingly.

2083 <Target> [Required]

2084 The <Target> element defines the applicability of a <Policy> to a set of **decision requests**.

2085 The <Target> element MAY be declared by the creator of the <Policy> element, or it MAY be
2086 computed from the <Target> elements of the referenced <Rule> elements either as an
2087 intersection or as a union.

2088 <VariableDefinition> [Any Number]
 2089 Common function definitions that can be referenced from anywhere in a *rule* where an
 2090 expression can be found.

2091 <Rule> [Any Number]
 2092 A sequence of *rules* that MUST be combined according to the RuleCombiningAlgId attribute.
 2093 *Rules* whose <Target> elements and conditions match the *decision request* MUST be
 2094 considered. *Rules* whose <Target> elements or conditions do not match the *decision request*
 2095 SHALL be ignored.

2096 <ObligationExpressions> [Optional]
 2097 A *conjunctive sequence* of *obligation* expressions which MUST be evaluated into *obligations*
 2098 by the PDP. The corresponding *obligations* MUST be fulfilled by the *PEP* in conjunction with the
 2099 *authorization decision*. See Section 7.18 for a description of how the set of *obligations* to be
 2100 returned by the *PDP* SHALL be determined. See section 7.2 about enforcement of *obligations*.

2101 <AdviceExpressions> [Optional]
 2102 A *conjunctive sequence* of *advice* expressions which MUST evaluated into *advice* by the *PDP*.
 2103 The corresponding *advice* provide supplementary information to the *PEP* in conjunction with the
 2104 *authorization decision*. See Section 7.18 for a description of how the set of *advice* to be
 2105 returned by the *PDP* SHALL be determined.

2106 5.15 Element <PolicyDefaults>

2107 The <PolicyDefaults> element SHALL specify default values that apply to the <Policy> element.

```
2108 <xs:element name="PolicyDefaults" type="xacml:DefaultsType"/>
2109 <xs:complexType name="DefaultsType">
2110   <xs:sequence>
2111     <xs:choice>
2112       <xs:element ref="xacml:XPathVersion" />
2113     </xs:choice>
2114   </xs:sequence>
2115 </xs:complexType>
```

2116 <PolicyDefaults> element is of DefaultsType complex type.

2117 The <PolicyDefaults> element contains the following elements:

2118 <XPathVersion> [Optional]

2119 Default XPath version.

2120 5.16 Element <CombinerParameters>

2121 The <CombinerParameters> element conveys parameters for a *policy-* or *rule-combining algorithm*.

2122 If multiple <CombinerParameters> elements occur within the same *policy* or *policy set*, they SHALL
 2123 be considered equal to one <CombinerParameters> element containing the concatenation of all the
 2124 sequences of <CombinerParameters> contained in all the aforementioned <CombinerParameters>
 2125 elements, such that the order of occurrence of the <CombinerParameters> elements is preserved in
 2126 the concatenation of the <CombinerParameter> elements.

2127 Note that none of the combining algorithms specified in XACML 3.0 is parameterized.

```
2128 <xs:element name="CombinerParameters" type="xacml:CombinerParametersType"/>
2129 <xs:complexType name="CombinerParametersType">
2130   <xs:sequence>
2131     <xs:element ref="xacml:CombinerParameter" minOccurs="0"
2132       maxOccurs="unbounded"/>
2133   </xs:sequence>
```

2134 `</xs:complexType>`

2135 The `<CombinerParameters>` element is of `CombinerParametersType` complex type.

2136 The `<CombinerParameters>` element contains the following elements:

2137 `<CombinerParameter>` [Any Number]

2138 A single parameter. See Section 5.17.

2139 Support for the `<CombinerParameters>` element is optional.

2140 5.17 Element `<CombinerParameter>`

2141 The `<CombinerParameter>` element conveys a single parameter for a *policy- or rule-combining algorithm*.

```
2143 <xs:element name="CombinerParameter" type="xacml:CombinerParameterType"/>
2144 <xs:complexType name="CombinerParameterType">
2145   <xs:sequence>
2146     <xs:element ref="xacml:AttributeValue"/>
2147   </xs:sequence>
2148   <xs:attribute name="ParameterName" type="xs:string" use="required"/>
2149 </xs:complexType>
```

2150 The `<CombinerParameter>` element is of `CombinerParameterType` complex type.

2151 The `<CombinerParameter>` element contains the following attributes:

2152 `ParameterName` [Required]

2153 The identifier of the parameter.

2154 `<AttributeValue>` [Required]

2155 The value of the parameter.

2156 Support for the `<CombinerParameter>` element is optional.

2157 5.18 Element `<RuleCombinerParameters>`

2158 The `<RuleCombinerParameters>` element conveys parameters associated with a particular *rule* within a *policy* for a *rule-combining algorithm*.

2160 Each `<RuleCombinerParameters>` element MUST be associated with a *rule* contained within the same *policy*. If multiple `<RuleCombinerParameters>` elements reference the same *rule*, they SHALL be considered equal to one `<RuleCombinerParameters>` element containing the concatenation of all the sequences of `<CombinerParameters>` contained in all the aforementioned `<RuleCombinerParameters>` elements, such that the order of occurrence of the `<RuleCombinerParameters>` elements is preserved in the concatenation of the `<CombinerParameter>` elements.

2167 Note that none of the *rule-combining algorithms* specified in XACML 3.0 is parameterized.

```
2168 <xs:element name="RuleCombinerParameters"
2169   type="xacml:RuleCombinerParametersType"/>
2170 <xs:complexType name="RuleCombinerParametersType">
2171   <xs:complexContent>
2172     <xs:extension base="xacml:CombinerParametersType">
2173       <xs:attribute name="RuleIdRef" type="xs:string"
2174         use="required"/>
2175     </xs:extension>
2176   </xs:complexContent>
2177 </xs:complexType>
```

2178 The `<RuleCombinerParameters>` element contains the following attribute:

2179 RuleIdRef [Required]
2180 The identifier of the <Rule> contained in the *policy*.
2181 Support for the <RuleCombinerParameters> element is optional, only if support for combiner
2182 parameters is not implemented.

2183 5.19 Element <PolicyCombinerParameters>

2184 The <PolicyCombinerParameters> element conveys parameters associated with a particular *policy*
2185 within a *policy set* for a *policy-combining algorithm*.

2186 Each <PolicyCombinerParameters> element MUST be associated with a *policy* contained within the
2187 same *policy set*. If multiple <PolicyCombinerParameters> elements reference the same *policy*,
2188 they SHALL be considered equal to one <PolicyCombinerParameters> element containing the
2189 concatenation of all the sequences of <CombinerParameters> contained in all the aforementioned
2190 <PolicyCombinerParameters> elements, such that the order of occurrence of the
2191 <PolicyCombinerParameters> elements is preserved in the concatenation of the
2192 <CombinerParameter> elements.

2193 Note that none of the *policy-combining algorithms* specified in XACML 3.0 is parameterized.

```
2194 <xs:element name="PolicyCombinerParameters"  
2195 type="xacml:PolicyCombinerParametersType"/>  
2196 <xs:complexType name="PolicyCombinerParametersType">  
2197   <xs:complexContent>  
2198     <xs:extension base="xacml:CombinerParametersType">  
2199       <xs:attribute name="PolicyIdRef" type="xs:anyURI"  
2200 use="required"/>  
2201     </xs:extension>  
2202   </xs:complexContent>  
2203 </xs:complexType>
```

2204 The <PolicyCombinerParameters> element is of PolicyCombinerParametersType complex
2205 type.

2206 The <PolicyCombinerParameters> element contains the following attribute:

2207 PolicyIdRef [Required]

2208 The identifier of a <Policy> or the value of a <PolicyIdReference> contained in the *policy*
2209 *set*.

2210 Support for the <PolicyCombinerParameters> element is optional, only if support for combiner
2211 parameters is not implemented.

2212 5.20 Element <PolicySetCombinerParameters>

2213 The <PolicySetCombinerParameters> element conveys parameters associated with a particular
2214 *policy set* within a *policy set* for a *policy-combining algorithm*.

2215 Each <PolicySetCombinerParameters> element MUST be associated with a *policy set* contained
2216 within the same *policy set*. If multiple <PolicySetCombinerParameters> elements reference the
2217 same *policy set*, they SHALL be considered equal to one <PolicySetCombinerParameters>
2218 element containing the concatenation of all the sequences of <CombinerParameters> contained in all
2219 the aforementioned <PolicySetCombinerParameters> elements, such that the order of occurrence
2220 of the <PolicySetCombinerParameters> elements is preserved in the concatenation of the
2221 <CombinerParameter> elements.

2222 Note that none of the *policy-combining algorithms* specified in XACML 3.0 is parameterized.

```
2223 <xs:element name="PolicySetCombinerParameters"  
2224 type="xacml:PolicySetCombinerParametersType"/>  
2225 <xs:complexType name="PolicySetCombinerParametersType">
```

2226
2227
2228
2229
2230
2231
2232

```
<xs:complexContent>
  <xs:extension base="xacml:CombinerParametersType">
    <xs:attribute name="PolicySetIdRef" type="xs:anyURI"
use="required"/>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

2233 The <PolicySetCombinerParameters> element is of PolicySetCombinerParametersType
2234 complex type.

2235 The <PolicySetCombinerParameters> element contains the following attribute:

2236 PolicySetIdRef [Required]

2237 The identifier of a <PolicySet> or the value of a <PolicySetIdReference> contained in the
2238 **policy set**.

2239 Support for the <PolicySetCombinerParameters> element is optional, only if support for combiner
2240 parameters is not implemented.

2241 5.21 Element <Rule>

2242 The <Rule> element SHALL define the individual **rules** in the **policy**. The main components of this
2243 element are the <Target>, <Condition>, <ObligationExpressions> and
2244 <AdviceExpressions> elements and the Effect attribute.

2245 A <Rule> element may be evaluated, in which case the evaluation procedure defined in Section 7.10
2246 SHALL be used.

2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258

```
<xs:element name="Rule" type="xacml:RuleType"/>
<xs:complexType name="RuleType">
  <xs:sequence>
    <xs:element ref="xacml:Description" minOccurs="0"/>
    <xs:element ref="xacml:Target" minOccurs="0"/>
    <xs:element ref="xacml:Condition" minOccurs="0"/>
    <xs:element ref="xacml:ObligationExpressions" minOccurs="0"/>
    <xs:element ref="xacml:AdviceExpressions" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="RuleId" type="xs:string" use="required"/>
  <xs:attribute name="Effect" type="xacml:EffectType" use="required"/>
</xs:complexType>
```

2259 The <Rule> element is of RuleType complex type.

2260 The <Rule> element contains the following attributes and elements:

2261 RuleId [Required]

2262 A string identifying this **rule**.

2263 Effect [Required]

2264 **Rule effect**. The value of this attribute is either "Permit" or "Deny".

2265 <Description> [Optional]

2266 A free-form description of the **rule**.

2267 <Target> [Optional]

2268 Identifies the set of **decision requests** that the <Rule> element is intended to evaluate. If this
2269 element is omitted, then the **target** for the <Rule> SHALL be defined by the <Target> element
2270 of the enclosing <Policy> element. See Section 7.7 for details.

2271 <Condition> [Optional]

2272 A **predicate** that MUST be satisfied for the **rule** to be assigned its Effect value.

2273 <ObligationExpressions> [Optional]

2274 A **conjunctive sequence** of **obligation** expressions which MUST be evaluated into **obligations**
2275 by the PDP. The corresponding **obligations** MUST be fulfilled by the **PEP** in conjunction with the
2276 **authorization decision**. See Section 7.18 for a description of how the set of **obligations** to be
2277 returned by the **PDP** SHALL be determined. See section 7.2 about enforcement of **obligations**.

2278 <AdviceExpressions> [Optional]

2279 A **conjunctive sequence** of **advice** expressions which MUST evaluated into **advice** by the **PDP**.
2280 The corresponding **advice** provide supplementary information to the **PEP** in conjunction with the
2281 **authorization decision**. See Section 7.18 for a description of how the set of **advice** to be
2282 returned by the **PDP** SHALL be determined.

2283 5.22 Simple type EffectType

2284 The EffectType simple type defines the values allowed for the Effect attribute of the <Rule> element
2285 and for the FulfillOn attribute of the <ObligationExpression> and <AdviceExpression>
2286 elements.

```
2287 <xs:simpleType name="EffectType">  
2288   <xs:restriction base="xs:string">  
2289     <xs:enumeration value="Permit"/>  
2290     <xs:enumeration value="Deny"/>  
2291   </xs:restriction>  
2292 </xs:simpleType>
```

2293 5.23 Element <VariableDefinition>

2294 The <VariableDefinition> element SHALL be used to define a value that can be referenced by a
2295 <VariableReference> element. The name supplied for its VariableId attribute SHALL NOT occur
2296 in the VariableId attribute of any other <VariableDefinition> element within the encompassing
2297 **policy**. The <VariableDefinition> element MAY contain undefined <VariableReference>
2298 elements, but if it does, a corresponding <VariableDefinition> element MUST be defined later in
2299 the encompassing **policy**. <VariableDefinition> elements MAY be grouped together or MAY be
2300 placed close to the reference in the encompassing **policy**. There MAY be zero or more references to
2301 each <VariableDefinition> element.

```
2302 <xs:element name="VariableDefinition" type="xacml:VariableDefinitionType"/>  
2303 <xs:complexType name="VariableDefinitionType">  
2304   <xs:sequence>  
2305     <xs:element ref="xacml:Expression"/>  
2306   </xs:sequence>  
2307   <xs:attribute name="VariableId" type="xs:string" use="required"/>  
2308 </xs:complexType>
```

2309 The <VariableDefinition> element is of VariableDefinitionType complex type. The
2310 <VariableDefinition> element has the following elements and attributes:

2311 <Expression> [Required]

2312 Any element of ExpressionType complex type.

2313 VariableId [Required]

2314 The name of the variable definition.

2315 5.24 Element <VariableReference>

2316 The <VariableReference> element is used to reference a value defined within the same
2317 encompassing <Policy> element. The <VariableReference> element SHALL refer to the
2318 <VariableDefinition> element by **identifier equality** on the value of their respective VariableId

2319 attributes. One and only one <VariableDefinition> MUST exist within the same encompassing
2320 <Policy> element to which the <VariableReference> refers. There MAY be zero or more
2321 <VariableReference> elements that refer to the same <VariableDefinition> element.

```
2322 <xs:element name="VariableReference" type="xacml:VariableReferenceType"  
2323 substitutionGroup="xacml:Expression"/>  
2324 <xs:complexType name="VariableReferenceType">  
2325 <xs:complexContent>  
2326 <xs:extension base="xacml:ExpressionType">  
2327 <xs:attribute name="VariableId" type="xs:string"  
2328 use="required"/>  
2329 </xs:extension>  
2330 </xs:complexContent>  
2331 </xs:complexType>
```

2332 The <VariableReference> element is of the VariableReferenceType complex type, which is of
2333 the ExpressionType complex type and is a member of the <Expression> element substitution group.
2334 The <VariableReference> element MAY appear any place where an <Expression> element occurs
2335 in the schema.

2336 The <VariableReference> element has the following attribute:

2337 VariableId [Required]

2338 The name used to refer to the value defined in a <VariableDefinition> element.

2339 5.25 Element <Expression>

2340 The <Expression> element is not used directly in a **policy**. The <Expression> element signifies that
2341 an element that extends the ExpressionType and is a member of the <Expression> element
2342 substitution group SHALL appear in its place.

```
2343 <xs:element name="Expression" type="xacml:ExpressionType" abstract="true"/>  
2344 <xs:complexType name="ExpressionType" abstract="true"/>
```

2345 The following elements are in the <Expression> element substitution group:

2346 <Apply>, <AttributeSelector>, <AttributeValue>, <Function>, <VariableReference> and
2347 <AttributeDesignator>.

2348 5.26 Element <Condition>

2349 The <Condition> element is a Boolean function over **attributes** or functions of **attributes**.

```
2350 <xs:element name="Condition" type="xacml:ConditionType"/>  
2351 <xs:complexType name="ConditionType">  
2352 <xs:sequence>  
2353 <xs:element ref="xacml:Expression"/>  
2354 </xs:sequence>  
2355 </xs:complexType>
```

2356 The <Condition> contains one <Expression> element, with the restriction that the <Expression>
2357 return data-type MUST be "http://www.w3.org/2001/XMLSchema#boolean". Evaluation of the
2358 <Condition> element is described in Section 7.9.

2359 5.27 Element <Apply>

2360 The <Apply> element denotes application of a function to its arguments, thus encoding a function call.
2361 The <Apply> element can be applied to any combination of the members of the <Expression>
2362 element substitution group. See Section 5.25.

```
2363 <xs:element name="Apply" type="xacml:ApplyType"  
2364 substitutionGroup="xacml:Expression"/>
```



```

2365 <xs:complexType name="ApplyType">
2366   <xs:complexContent>
2367     <xs:extension base="xacml:ExpressionType">
2368       <xs:sequence>
2369         <xs:element ref="xacml:Description" minOccurs="0"/>
2370         <xs:element ref="xacml:Expression" minOccurs="0"
2371           maxOccurs="unbounded"/>
2372       </xs:sequence>
2373       <xs:attribute name="FunctionId" type="xs:anyURI"
2374         use="required"/>
2375     </xs:extension>
2376   </xs:complexContent>
2377 </xs:complexType>

```

2378 The <Apply> element is of ApplyType complex type.

2379 The <Apply> element contains the following attributes and elements:

2380 FunctionId [Required]

2381 The identifier of the function to be applied to the arguments. XACML-defined functions are
2382 described in Appendix A.3.

2383 <Description> [Optional]

2384 A free-form description of the <Apply> element.

2385 <Expression> [Optional]

2386 Arguments to the function, which may include other functions.

2387 5.28 Element <Function>

2388 The <Function> element SHALL be used to name a function as an argument to the function defined by
2389 the parent <Apply> element.

```

2390 <xs:element name="Function" type="xacml:FunctionType"
2391 substitutionGroup="xacml:Expression"/>
2392 <xs:complexType name="FunctionType">
2393   <xs:complexContent>
2394     <xs:extension base="xacml:ExpressionType">
2395       <xs:attribute name="FunctionId" type="xs:anyURI"
2396         use="required"/>
2397     </xs:extension>
2398   </xs:complexContent>
2399 </xs:complexType>

```

2400 The <Function> element is of FunctionType complex type.

2401 The <Function> element contains the following attribute:

2402 FunctionId [Required]

2403 The identifier of the function.

2404 5.29 Element <AttributeDesignator>

2405 The <AttributeDesignator> element retrieves a **bag** of values for a **named attribute** from the
2406 request **context**. A **named attribute** SHALL be considered present if there is at least one **attribute** that
2407 matches the criteria set out below.

2408 The <AttributeDesignator> element SHALL return a **bag** containing all the **attribute** values that are
2409 matched by the **named attribute**. In the event that no matching **attribute** is present in the **context**, the
2410 MustBePresent attribute governs whether this element returns an empty **bag** or "Indeterminate". See
2411 Section 7.3.5.

2412 The <AttributeDesignator> MAY appear in the <Match> element and MAY be passed to the
2413 <Apply> element as an argument.

2414 The <AttributeDesignator> element is of the AttributeDesignatorType complex type.

```
2415 <xs:element name="AttributeDesignator" type="xacml:AttributeDesignatorType"  
2416 substitutionGroup="xacml:Expression"/>  
2417 <xs:complexType name="AttributeDesignatorType">  
2418 <xs:complexContent>  
2419 <xs:extension base="xacml:ExpressionType">  
2420 <xs:attribute name="Category" type="xs:anyURI"  
2421 use="required"/>  
2422 <xs:attribute name="AttributeId" type="xs:anyURI"  
2423 use="required"/>  
2424 <xs:attribute name="DataType" type="xs:anyURI"  
2425 use="required"/>  
2426 <xs:attribute name="Issuer" type="xs:string" use="optional"/>  
2427 <xs:attribute name="MustBePresent" type="xs:boolean"  
2428 use="required"/>  
2429 </xs:extension>  
2430 </xs:complexContent>  
2431 </xs:complexType>
```

2432 A **named attribute** SHALL match an **attribute** if the values of their respective Category,
2433 AttributeId, DataType and Issuer attributes match. The attribute designator's Category MUST
2434 match, by **identifier equality**, the Category of the <Attributes> element in which the **attribute** is
2435 present. The attribute designator's AttributeId MUST match, by **identifier equality**, the
2436 AttributeId of the attribute. The attribute designator's DataType MUST match, by **identifier**
2437 **equality**, the DataType of the same **attribute**.

2438 If the Issuer attribute is present in the attribute designator, then it MUST match, using the
2439 "urn:oasis:names:tc:xacml:1.0:function:string-equal" function, the Issuer of the same **attribute**. If the
2440 Issuer is not present in the attribute designator, then the matching of the **attribute** to the **named**
2441 **attribute** SHALL be governed by AttributeId and DataType attributes alone.

2442 The <AttributeDesignatorType> contains the following attributes:

2443 Category [Required]

2444 This attribute SHALL specify the Category with which to match the **attribute**.

2445 AttributeId [Required]

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

2447 DataType [Required]

2448 The **bag** returned by the <AttributeDesignator> element SHALL contain values of this data-
2449 type.

2450 Issuer [Optional]

2451 This attribute, if supplied, SHALL specify the Issuer with which to match the **attribute**.

2452 MustBePresent [Required]

2453 This attribute governs whether the element returns "Indeterminate" or an empty **bag** in the event
2454 the **named attribute** is absent from the request **context**. See Section 7.3.5. Also see Sections
2455 7.19.2 and 7.19.3.

2456 5.30 Element <AttributeSelector>

2457 The <AttributeSelector> element produces a **bag** of unnamed and uncategorized **attribute** values.
2458 The values shall be constructed from the node(s) selected by applying the XPath expression given by the
2459 element's Path attribute to the XML content indicated by the element's Category attribute. Support for
2460 the <AttributeSelector> element is OPTIONAL.

2461 See section 7.3.7 for details of <AttributeSelector> evaluation.

```
2462 <xs:element name="AttributeSelector" type="xacml:AttributeSelectorType"  
2463 substitutionGroup="xacml:Expression"/>  
2464 <xs:complexType name="AttributeSelectorType">  
2465   <xs:complexContent>  
2466     <xs:extension base="xacml:ExpressionType">  
2467       <xs:attribute name="Category" type="xs:anyURI"  
2468         use="required"/>  
2469       <xs:attribute name="ContextSelectorId" type="xs:anyURI"  
2470         use="optional"/>  
2471       <xs:attribute name="Path" type="xs:string"  
2472         use="required"/>  
2473       <xs:attribute name="DataType" type="xs:anyURI"  
2474         use="required"/>  
2475       <xs:attribute name="MustBePresent" type="xs:boolean"  
2476         use="required"/>  
2477     </xs:extension>  
2478   </xs:complexContent>  
2479 </xs:complexType>
```

2480 The <AttributeSelector> element is of AttributeSelectorType complex type.

2481 The <AttributeSelector> element has the following attributes:

2482 Category [Required]

2483 This attribute SHALL specify the **attributes** category of the <Content> element containing the
2484 XML from which nodes will be selected. It also indicates the **attributes** category containing the
2485 applicable ContextSelectorId attribute, if the element includes a ContextSelectorId xml
2486 attribute.

2487 ContextSelectorId [Optional]

2488 This attribute refers to the **attribute** (by its AttributeId) in the request **context** in the category
2489 given by the Category attribute. The referenced **attribute** MUST have data type
2490 urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression, and must select a single node in the
2491 <Content> element. The XPathCategory attribute of the referenced **attribute** MUST be equal
2492 to the Category attribute of the **attribute selector**.

2493 Path [Required]

2494 This attribute SHALL contain an XPath expression to be evaluated against the specified XML
2495 content. See Section 7.3.7 for details of the XPath evaluation during <AttributeSelector>
2496 processing. The namespace context for the value of the Path attribute is given by the [in-scope
2497 namespaces] [INFOSET] of the <AttributeSelector> element.

2498 DataType [Required]

2499 The attribute specifies the datatype of the values returned from the evaluation of this
2500 <AttributeSelector> element.

2501 MustBePresent [Required]

2502 This attribute governs whether the element returns "Indeterminate" or an empty **bag** in the event
2503 that the attributes category specified by the Category attribute does not exist in the request
2504 context, or the attributes category does exist but it does not have a <Content> child element, or
2505 the <Content> element does exist but the XPath expression selects no node. See Section 7.3.5.
2506 Also see Sections 7.19.2 and 7.19.3.

2507 5.31 Element <AttributeValue>

2508 The <AttributeValue> element SHALL contain a literal **attribute** value.

```

2509 <xs:element name="AttributeValue" type="xacml:AttributeValueType"
2510 substitutionGroup="xacml:Expression"/>
2511 <xs:complexType name="AttributeValueType" mixed="true">
2512   <xs:complexContent mixed="true">
2513     <xs:extension base="xacml:ExpressionType">
2514       <xs:sequence>
2515         <xs:any namespace="##any" processContents="lax"
2516           minOccurs="0" maxOccurs="unbounded"/>
2517       </xs:sequence>
2518       <xs:attribute name="DataType" type="xs:anyURI"
2519         use="required"/>
2520       <xs:anyAttribute namespace="##any" processContents="lax"/>
2521     </xs:extension>
2522   </xs:complexContent>
2523 </xs:complexType>

```

2524 The <AttributeValue> element is of AttributeValueType complex type.

2525 The <AttributeValue> element has the following attributes:

2526 DataType [Required]

2527 The data-type of the *attribute* value.

2528 5.32 Element <Obligations>

2529 The <Obligations> element SHALL contain a set of <Obligation> elements.

```

2530 <xs:element name="Obligations" type="xacml:ObligationsType"/>
2531 <xs:complexType name="ObligationsType">
2532   <xs:sequence>
2533     <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>
2534   </xs:sequence>
2535 </xs:complexType>

```

2536 The <Obligations> element is of ObligationsType complexType.

2537 The <Obligations> element contains the following element:

2538 <Obligation> [One to Many]

2539 A sequence of *obligations*. See Section 5.34.

2540 5.33 Element <AssociatedAdvice>

2541 The <AssociatedAdvice> element SHALL contain a set of <Advice> elements.

```

2542 <xs:element name="AssociatedAdvice" type="xacml:AssociatedAdviceType"/>
2543 <xs:complexType name="AssociatedAdviceType">
2544   <xs:sequence>
2545     <xs:element ref="xacml:Advice" maxOccurs="unbounded"/>
2546   </xs:sequence>
2547 </xs:complexType>

```

2548 The <AssociatedAdvice> element is of AssociatedAdviceType complexType.

2549 The <AssociatedAdvice> element contains the following element:

2550 <Advice> [One to Many]

2551 A sequence of *advice*. See Section 5.35.

2552 5.34 Element <Obligation>

2553 The <Obligation> element SHALL contain an identifier for the *obligation* and a set of *attributes* that
2554 form arguments of the action defined by the *obligation*.

```

2555 <xs:element name="Obligation" type="xacml:ObligationType"/>
2556 <xs:complexType name="ObligationType">
2557   <xs:sequence>
2558     <xs:element ref="xacml:AttributeAssignment" minOccurs="0"
2559       maxOccurs="unbounded"/>
2560   </xs:sequence>
2561   <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
2562 </xs:complexType>

```

2563 The <Obligation> element is of ObligationType complexType. See Section 7.18 for a description
 2564 of how the set of **obligations** to be returned by the **PDP** is determined.

2565 The <Obligation> element contains the following elements and attributes:

2566 ObligationId [Required]

2567 **Obligation** identifier. The value of the **obligation** identifier SHALL be interpreted by the **PEP**.

2568 <AttributeAssignment> [Optional]

2569 **Obligation** arguments assignment. The values of the **obligation** arguments SHALL be
 2570 interpreted by the **PEP**.

2571 5.35 Element <Advice>

2572 The <Advice> element SHALL contain an identifier for the **advice** and a set of **attributes** that form
 2573 arguments of the supplemental information defined by the **advice**.

```

2574 <xs:element name="Advice" type="xacml:AdviceType"/>
2575 <xs:complexType name="AdviceType">
2576   <xs:sequence>
2577     <xs:element ref="xacml:AttributeAssignment" minOccurs="0"
2578       maxOccurs="unbounded"/>
2579   </xs:sequence>
2580   <xs:attribute name="AdviceId" type="xs:anyURI" use="required"/>
2581 </xs:complexType>

```

2582 The <Advice> element is of AdviceType complexType. See Section 7.18 for a description of how the
 2583 set of **advice** to be returned by the **PDP** is determined.

2584 The <Advice> element contains the following elements and attributes:

2585 AdviceId [Required]

2586 **Advice** identifier. The value of the **advice** identifier MAY be interpreted by the **PEP**.

2587 <AttributeAssignment> [Optional]

2588 **Advice** arguments assignment. The values of the **advice** arguments MAY be interpreted by the
 2589 **PEP**.

2590 5.36 Element <AttributeAssignment>

2591 The <AttributeAssignment> element is used for including arguments in **obligation** and **advice**
 2592 expressions. It SHALL contain an AttributeId and the corresponding **attribute** value, by extending
 2593 the AttributeValueType type definition. The <AttributeAssignment> element MAY be used in
 2594 any way that is consistent with the schema syntax, which is a sequence of <xs:any> elements. The
 2595 value specified SHALL be understood by the **PEP**, but it is not further specified by XACML. See Section
 2596 7.18. Section 4.2.4.3 provides a number of examples of arguments included in **obligation** expressions.

```

2597 <xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
2598 <xs:complexType name="AttributeAssignmentType" mixed="true">
2599   <xs:complexContent>
2600     <xs:extension base="xacml:AttributeValueType">
2601       <xs:attribute name="AttributeId" type="xs:anyURI"
2602         use="required"/>

```

```

2603         <xs:attribute name="Category" type="xs:anyURI"
2604             use="optional" />
2605         <xs:attribute name="Issuer" type="xs:string" use="optional" />
2606     </xs:extension>
2607 </xs:complexContent>
2608 </xs:complexType>

```

2609 The <AttributeAssignment> element is of AttributeAssignmentType complex type.

2610 The <AttributeAssignment> element contains the following attributes:

2611 AttributeId [Required]

2612 The **attribute** Identifier.

2613 Category [Optional]

2614 An optional category of the **attribute**. If this attribute is missing, the **attribute** has no category.

2615 The **PEP** SHALL interpret the significance and meaning of any Category attribute. Non-

2616 normative note: an expected use of the category is to disambiguate **attributes** which are relayed
 2617 from the request.

2618 Issuer [Optional]

2619 An optional issuer of the **attribute**. If this attribute is missing, the **attribute** has no issuer. The

2620 **PEP** SHALL interpret the significance and meaning of any Issuer attribute. Non-normative note:
 2621 an expected use of the issuer is to disambiguate **attributes** which are relayed from the request.

2622 5.37 Element <ObligationExpressions>

2623 The <ObligationExpressions> element SHALL contain a set of <ObligationExpression>
 2624 elements.

```

2625 <xs:element name="ObligationExpressions"
2626     type="xacml:ObligationExpressionsType" />
2627 <xs:complexType name="ObligationExpressionsType">
2628     <xs:sequence>
2629         <xs:element ref="xacml:ObligationExpression" maxOccurs="unbounded" />
2630     </xs:sequence>
2631 </xs:complexType>

```

2632 The <ObligationExpressions> element is of ObligationExpressionsType complexType.

2633 The <ObligationExpressions> element contains the following element:

2634 <ObligationExpression> [One to Many]

2635 A sequence of **obligations** expressions. See Section 5.39.

2636 5.38 Element <AdviceExpressions>

2637 The <AdviceExpressions> element SHALL contain a set of <AdviceExpression> elements.

```

2638 <xs:element name="AdviceExpressions" type="xacml:AdviceExpressionsType" />
2639 <xs:complexType name="AdviceExpressionsType">
2640     <xs:sequence>
2641         <xs:element ref="xacml:AdviceExpression" maxOccurs="unbounded" />
2642     </xs:sequence>
2643 </xs:complexType>

```

2644 The <AdviceExpressions> element is of AdviceExpressionsType complexType.

2645 The <AdviceExpressions> element contains the following element:

2646 <AdviceExpression> [One to Many]

2647 A sequence of **advice** expressions. See Section 5.40.

2648 5.39 Element <ObligationExpression>

2649 The <ObligationExpression> element evaluates to an **obligation** and SHALL contain an identifier
2650 for an **obligation** and a set of expressions that form arguments of the action defined by the **obligation**.
2651 The FulfillOn attribute SHALL indicate the **effect** for which this **obligation** must be fulfilled by the
2652 **PEP**.

```
2653 <xs:element name="ObligationExpression"  
2654     type="xacml:ObligationExpressionType"/>  
2655 <xs:complexType name="ObligationExpressionType">  
2656   <xs:sequence>  
2657     <xs:element ref="xacml:AttributeAssignmentExpression" minOccurs="0"  
2658       maxOccurs="unbounded"/>  
2659   </xs:sequence>  
2660   <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>  
2661   <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>  
2662 </xs:complexType>
```

2663 The <ObligationExpression> element is of ObligationExpressionType complexType. See
2664 Section 7.18 for a description of how the set of **obligations** to be returned by the **PDP** is determined.

2665 The <ObligationExpression> element contains the following elements and attributes:

2666 ObligationId [Required]

2667 **Obligation** identifier. The value of the **obligation** identifier SHALL be interpreted by the **PEP**.

2668 FulfillOn [Required]

2669 The **effect** for which this **obligation** must be fulfilled by the **PEP**.

2670 <AttributeAssignmentExpression> [Optional]

2671 **Obligation** arguments in the form of expressions. The expressions SHALL be evaluated by the
2672 PDP to constant <AttributeValue> elements or **bags**, which shall be the attribute
2673 assignments in the <Obligation> returned to the PEP. If an
2674 <AttributeAssignmentExpression> evaluates to an atomic **attribute** value, then there
2675 MUST be one resulting <AttributeAssignment> which MUST contain this single **attribute**
2676 value. If the <AttributeAssignmentExpression> evaluates to a **bag**, then there MUST be a
2677 resulting <AttributeAssignment> for each of the values in the **bag**. If the **bag** is empty, there
2678 shall be no <AttributeAssignment> from this <AttributeAssignmentExpression>. The
2679 values of the **obligation** arguments SHALL be interpreted by the **PEP**.

2680 5.40 Element <AdviceExpression>

2681 The <AdviceExpression> element evaluates to an **advice** and SHALL contain an identifier for an
2682 **advice** and a set of expressions that form arguments of the supplemental information defined by the
2683 **advice**. The AppliesTo attribute SHALL indicate the **effect** for which this **advice** must be provided to
2684 the **PEP**.

```
2685 <xs:element name="AdviceExpression" type="xacml:AdviceExpressionType"/>  
2686 <xs:complexType name="AdviceExpressionType">  
2687   <xs:sequence>  
2688     <xs:element ref="xacml:AttributeAssignmentExpression" minOccurs="0"  
2689       maxOccurs="unbounded"/>  
2690   </xs:sequence>  
2691   <xs:attribute name="AdviceId" type="xs:anyURI" use="required"/>  
2692   <xs:attribute name="AppliesTo" type="xacml:EffectType" use="required"/>  
2693 </xs:complexType>
```

2694 The <AdviceExpression> element is of AdviceExpressionType complexType. See Section 7.18
2695 for a description of how the set of **advice** to be returned by the **PDP** is determined.

2696 The <AdviceExpression> element contains the following elements and attributes:

2697 AdviceId [Required]
 2698 **Advice** identifier. The value of the **advice** identifier MAY be interpreted by the **PEP**.
 2699 AppliesTo [Required]
 2700 The **effect** for which this **advice** must be provided to the **PEP**.
 2701 <AttributeAssignmentExpression> [Optional]
 2702 **Advice** arguments in the form of expressions. The expressions SHALL be evaluated by the PDP
 2703 to constant <AttributeValue> elements or **bags**, which shall be the attribute assignments in
 2704 the <Advice> returned to the PEP. If an <AttributeAssignmentExpression> evaluates to
 2705 an atomic **attribute** value, then there MUST be one resulting <AttributeAssignment> which
 2706 MUST contain this single **attribute** value. If the <AttributeAssignmentExpression>
 2707 evaluates to a **bag**, then there MUST be a resulting <AttributeAssignment> for each of the
 2708 values in the **bag**. If the **bag** is empty, there shall be no <AttributeAssignment> from this
 2709 <AttributeAssignmentExpression>. The values of the **advice** arguments MAY be
 2710 interpreted by the **PEP**.

2711 5.41 Element <AttributeAssignmentExpression>

2712 The <AttributeAssignmentExpression> element is used for including arguments in **obligations**
 2713 and **advice**. It SHALL contain an AttributeId and an expression which SHALL be evaluated into the
 2714 corresponding **attribute** value. The value specified SHALL be understood by the **PEP**, but it is not further
 2715 specified by XACML. See Section 7.18. Section 4.2.4.3 provides a number of examples of arguments
 2716 included in **obligations**.

```

2717 <xs:element name="AttributeAssignmentExpression"
2718   type="xacml:AttributeAssignmentExpressionType"/>
2719 <xs:complexType name="AttributeAssignmentExpressionType">
2720   <xs:sequence>
2721     <xs:element ref="xacml:Expression"/>
2722   </xs:sequence>
2723   <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
2724   <xs:attribute name="Category" type="xs:anyURI" use="optional"/>
2725   <xs:attribute name="Issuer" type="xs:string" use="optional"/>
2726 </xs:complexType>
  
```

2727 The <AttributeAssignmentExpression> element is of AttributeAssignmentExpressionType
 2728 complex type.

2729 The <AttributeAssignmentExpression> element contains the following attributes:

2730 <Expression> [Required]

2731 The expression which evaluates to a constant **attribute** value or a bag of zero or more attribute
 2732 values. See section 5.25.

2733 AttributeId [Required]

2734 The **attribute** identifier. The value of the AttributeId attribute in the resulting
 2735 <AttributeAssignment> element MUST be equal to this value.

2736 Category [Optional]

2737 An optional category of the **attribute**. If this attribute is missing, the **attribute** has no category.
 2738 The value of the Category attribute in the resulting <AttributeAssignment> element MUST be
 2739 equal to this value.

2740 Issuer [Optional]

2741 An optional issuer of the **attribute**. If this attribute is missing, the **attribute** has no issuer. The
 2742 value of the Issuer attribute in the resulting <AttributeAssignment> element MUST be equal to
 2743 this value.

2744 5.42 Element <Request>

2745 The <Request> element is an abstraction layer used by the **policy** language. For simplicity of
2746 expression, this document describes **policy** evaluation in terms of operations on the **context**. However a
2747 conforming **PDP** is not required to actually instantiate the **context** in the form of an XML document. But,
2748 any system conforming to the XACML specification MUST produce exactly the same **authorization**
2749 **decisions** as if all the inputs had been transformed into the form of an <Request> element.

```
2750 <xs:element name="Request" type="xacml:RequestType"/>
2751 <xs:complexType name="RequestType">
2752   <xs:sequence>
2753     <xs:element ref="xacml:RequestDefaults" minOccurs="0"/>
2754     <xs:element ref="xacml:Attributes" maxOccurs="unbounded"/>
2755     <xs:element ref="xacml:MultiRequests" minOccurs="0"/>
2756   </xs:sequence>
2757   <xs:attribute name="ReturnPolicyIdList" type="xs:boolean" use="required"/>
2758   <xs:attribute name="CombinedDecision" type="xs:boolean" use="required" />
2759 </xs:complexType>
```

2760 The <Request> element is of RequestType complex type.

2761 The <Request> element contains the following elements and attributes:

2762 ReturnPolicyIdList [Required]

2763 This attribute is used to request that the **PDP** return a list of all fully applicable **policies** and
2764 **policy sets** which were used in the decision as a part of the decision response.

2765 CombinedDecision [Required]

2766 This attribute is used to request that the **PDP** combines multiple decisions into a single decision.
2767 The use of this attribute is specified in [Multi]. If the **PDP** does not implement the relevant
2768 functionality in [Multi], then the **PDP** must return an Indeterminate with a status code of
2769 urn:oasis:names:tc:xacml:1.0:status:processing-error if it receives a request with this attribute set
2770 to "true".

2771 <RequestDefaults> [Optional]

2772 Contains default values for the request, such as XPath version. See section 5.43.

2773 <Attributes> [One to Many]

2774 Specifies information about **attributes** of the request **context** by listing a sequence of
2775 <Attribute> elements associated with an **attribute** category. One or more <Attributes>
2776 elements are allowed. Different <Attributes> elements with different categories are used to
2777 represent information about the **subject**, **resource**, **action**, **environment** or other categories of
2778 the **access** request.

2779 The <Request> element contains <Attributes> elements. There may be multiple
2780 <Attributes> elements with the same Category attribute if the **PDP** implements the multiple
2781 decision profile, see [Multi]. Under other conditions, it is a syntax error if there are multiple
2782 <Attributes> elements with the same Category (see Section 7.19.2 for error codes).

2783 <MultiRequests> [Optional]

2784 Lists multiple **request contexts** by references to the <Attributes> elements. Implementation
2785 of this element is optional. The semantics of this element is defined in [Multi]. If the
2786 implementation does not implement this element, it MUST return an Indeterminate result if it
2787 encounters this element. See section 5.50.

2788 5.43 Element <RequestDefaults>

2789 The <RequestDefaults> element SHALL specify default values that apply to the <Request> element.

```
2790 <xs:element name="RequestDefaults" type="xacml:RequestDefaultsType"/>
```

```

2791 <xs:complexType name="RequestDefaultsType">
2792   <xs:sequence>
2793     <xs:choice>
2794       <xs:element ref="xacml:XPathVersion"/>
2795     </xs:choice>
2796   </xs:sequence>
2797 </xs:complexType>

```

2798 <RequestDefaults> element is of RequestDefaultsType complex type.

2799 The <RequestDefaults> element contains the following elements:

2800 <XPathVersion> [Optional]

2801 Default XPath version for XPath expressions occurring in the request.

2802 5.44 Element <Attributes>

2803 The <Attributes> element specifies **attributes** of a **subject**, **resource**, **action**, **environment** or
 2804 another category by listing a sequence of <Attribute> elements associated with the category.

```

2805 <xs:element name="Attributes" type="xacml:AttributesType"/>
2806 <xs:complexType name="AttributesType">
2807   <xs:sequence>
2808     <xs:element ref="xacml:Content" minOccurs="0"/>
2809     <xs:element ref="xacml:Attribute" minOccurs="0"
2810       maxOccurs="unbounded"/>
2811   </xs:sequence>
2812   <xs:attribute name="Category" type="xs:anyURI" use="required"/>
2813   <xs:attribute ref="xml:id" use="optional"/>
2814 </xs:complexType>

```

2815 The <Attributes> element is of AttributesType complex type.

2816 The <Attributes> element contains the following elements and attributes:

2817 Category [Required]

2818 This attribute indicates which **attribute** category the contained **attributes** belong to. The
 2819 Category attribute is used to differentiate between **attributes** of **subject**, **resource**, **action**,
 2820 **environment** or other categories.

2821 xml:id [Optional]

2822 This attribute provides a unique identifier for this <Attributes> element. See [XMLid] It is
 2823 primarily intended to be referenced in multiple requests. See [Multi].

2824 <Content> [Optional]

2825 Specifies additional sources of **attributes** in free form XML document format which can be
 2826 referenced using <AttributeSelector> elements.

2827 <Attribute> [Any Number]

2828 A sequence of **attributes** that apply to the category of the request.

2829 5.45 Element <Content>

2830 The <Content> element is a notional placeholder for additional **attributes**, typically the content of the
 2831 **resource**.

```

2832 <xs:element name="Content" type="xacml:ContentType"/>
2833 <xs:complexType name="ContentType" mixed="true">
2834   <xs:sequence>
2835     <xs:any namespace="##any" processContents="lax"/>
2836   </xs:sequence>
2837 </xs:complexType>

```

- 2838 The <Content> element is of Contentype complex type.
2839 The <Content> element has exactly one arbitrary type child element.

2840 5.46 Element <Attribute>

- 2841 The <Attribute> element is the central abstraction of the request **context**. It contains **attribute** meta-
2842 data and one or more **attribute** values. The **attribute** meta-data comprises the **attribute** identifier and
2843 the **attribute** issuer. <AttributeDesignator> elements in the **policy** MAY refer to **attributes** by
2844 means of this meta-data.

```
2845 <xs:element name="Attribute" type="xacml:AttributeType"/>  
2846 <xs:complexType name="AttributeType">  
2847   <xs:sequence>  
2848     <xs:element ref="xacml:AttributeValue" maxOccurs="unbounded"/>  
2849   </xs:sequence>  
2850   <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>  
2851   <xs:attribute name="Issuer" type="xs:string" use="optional"/>  
2852   <xs:attribute name="IncludeInResult" type="xs:boolean" use="required"/>  
2853 </xs:complexType>
```

- 2854 The <Attribute> element is of AttributeType complex type.
2855 The <Attribute> element contains the following attributes and elements:
2856 AttributeId [Required]
2857 The **Attribute** identifier. A number of identifiers are reserved by XACML to denote commonly
2858 used **attributes**. See Appendix Appendix B.
2859 Issuer [Optional]
2860 The **Attribute** issuer. For example, this attribute value MAY be an x500Name that binds to a
2861 public key, or it may be some other identifier exchanged out-of-band by issuing and relying
2862 parties.
2863 IncludeInResult [Default: false]
2864 Whether to include this **attribute** in the result. This is useful to correlate requests with their
2865 responses in case of multiple requests.
2866 <AttributeValue> [One to Many]
2867 One or more **attribute** values. Each **attribute** value MAY have contents that are empty, occur
2868 once or occur multiple times.

2869 5.47 Element <Response>

- 2870 The <Response> element is an abstraction layer used by the **policy** language. Any proprietary system
2871 using the XACML specification MUST transform an XACML **context** <Response> element into the form
2872 of its **authorization decision**.
2873 The <Response> element encapsulates the **authorization decision** produced by the **PDP**. It includes a
2874 sequence of one or more results, with one <Result> element per requested **resource**. Multiple results
2875 MAY be returned by some implementations, in particular those that support the XACML Profile for
2876 Requests for Multiple Resources [**Multi**]. Support for multiple results is OPTIONAL.

```
2877 <xs:element name="Response" type="xacml:ResponseType"/>  
2878 <xs:complexType name="ResponseType">  
2879   <xs:sequence>  
2880     <xs:element ref="xacml:Result" maxOccurs="unbounded"/>  
2881   </xs:sequence>  
2882 </xs:complexType>
```

- 2883 The <Response> element is of ResponseType complex type.

2884 The <Response> element contains the following elements:

2885 <Result> [One to Many]

2886 An **authorization decision** result. See Section 5.48.

2887 5.48 Element <Result>

2888 The <Result> element represents an **authorization decision** result. It MAY include a set of
2889 **obligations** that MUST be fulfilled by the **PEP**. If the **PEP** does not understand or cannot fulfill an
2890 **obligation**, then the action of the PEP is determined by its bias, see section 7.1. It MAY include a set of
2891 **advice** with supplemental information which MAY be safely ignored by the **PEP**.

```
2892 <xs:complexType name="ResultType">  
2893   <xs:sequence>  
2894     <xs:element ref="xacml:Decision"/>  
2895     <xs:element ref="xacml:Status" minOccurs="0"/>  
2896     <xs:element ref="xacml:Obligations" minOccurs="0"/>  
2897     <xs:element ref="xacml:AssociatedAdvice" minOccurs="0"/>  
2898     <xs:element ref="xacml:Attributes" minOccurs="0"  
2899       maxOccurs="unbounded"/>  
2900     <xs:element ref="xacml:PolicyIdentifierList" minOccurs="0"/>  
2901   </xs:sequence>  
2902 </xs:complexType>
```

2903 The <Result> element is of ResultType complex type.

2904 The <Result> element contains the following attributes and elements:

2905 <Decision> [Required]

2906 The **authorization decision**: “Permit”, “Deny”, “Indeterminate” or “NotApplicable”.

2907 <Status> [Optional]

2908 Indicates whether errors occurred during evaluation of the **decision request**, and optionally,
2909 information about those errors. If the <Response> element contains <Result> elements whose
2910 <Status> elements are all identical, and the <Response> element is contained in a protocol
2911 wrapper that can convey status information, then the common status information MAY be placed
2912 in the protocol wrapper and this <Status> element MAY be omitted from all <Result>
2913 elements.

2914 <Obligations> [Optional]

2915 A list of **obligations** that MUST be fulfilled by the **PEP**. If the **PEP** does not understand or cannot
2916 fulfill an **obligation**, then the action of the PEP is determined by its bias, see section 7.2. See
2917 Section 7.18 for a description of how the set of **obligations** to be returned by the **PDP** is
2918 determined.

2919 <AssociatedAdvice> [Optional]

2920 A list of **advice** that provide supplemental information to the **PEP**. If the **PEP** does not
2921 understand an **advice**, the PEP may safely ignore the **advice**. See Section 7.18 for a description
2922 of how the set of **advice** to be returned by the **PDP** is determined.

2923 <Attributes> [Optional]

2924 A list of **attributes** that were part of the request. The choice of which **attributes** are included here
2925 is made with the IncludeInResult attribute of the <Attribute> elements of the request. See
2926 section 5.46.

2927 <PolicyIdentifierList> [Optional]

2928 If the ReturnPolicyIdList attribute in the <Request> is true (see section 5.42), a **PDP** that
2929 implements this optional feature MUST return a list which includes the identifiers of all **policies**
2930 which were found to be fully applicable, whether or not the <Effect> (after rule combining) was
2931 the same or different from the <Decision>. The list MAY include the identifiers of other policies

2932 which are currently in force, as long as no policies required for the decision are omitted. A **PDP**
2933 MAY satisfy this requirement by including all policies currently in force, or by including all policies
2934 which were evaluated in making the decision, or by including all policies which did not evaluate to
2935 “NotApplicable”, or by any other algorithm which does not omit any policies which contributed to
2936 the decision. However, a decision which returns “NotApplicable” MUST return an empty list.

2937 5.49 Element <PolicyIdentifierList>

2938 The <PolicyIdentifierList> element contains a list of **policy** and **policy set** identifiers of **policies**
2939 which have been applicable to a request. The list is unordered.

```
2940 <xs:element name="PolicyIdentifierList"  
2941   type="xacml:PolicyIdentifierListType"/>  
2942 <xs:complexType name="PolicyIdentifierListType">  
2943   <xs:choice minOccurs="0" maxOccurs="unbounded">  
2944     <xs:element ref="xacml:PolicyIdReference"/>  
2945     <xs:element ref="xacml:PolicySetIdReference"/>  
2946   </xs:choice>  
2947 </xs:complexType>
```

2948 The <PolicyIdentifierList> element is of PolicyIdentifierListType complex type.

2949 The <PolicyIdentifierList> element contains the following elements.

2950 <PolicyIdReference> [Any number]

2951 The identifier and version of a **policy** which was applicable to the request. See section 5.11. The
2952 <PolicyIdReference> element MUST use the Version attribute to specify the version and
2953 MUST NOT use the LatestVersion or EarliestVersion attributes.

2954 <PolicySetIdReference> [Any number]

2955 The identifier and version of a **policy set** which was applicable to the request. See section 5.10.
2956 The <PolicySetIdReference> element MUST use the Version attribute to specify the
2957 version and MUST NOT use the LatestVersion or EarliestVersion attributes.

2958 5.50 Element <MultiRequests>

2959 The <MultiRequests> element contains a list of requests by reference to <Attributes> elements in
2960 the enclosing <Request> element. The semantics of this element are defined in [Multi]. Support for this
2961 element is optional. If an implementation does not support this element, but receives it, the
2962 implementation MUST generate an “Indeterminate” response.

```
2963 <xs:element name="MultiRequests" type="xacml:MultiRequestsType"/>  
2964 <xs:complexType name="MultiRequestsType">  
2965   <xs:sequence>  
2966     <xs:element ref="xacml:RequestReference" maxOccurs="unbounded"/>  
2967   </xs:sequence>  
2968 </xs:complexType>
```

2969 The <MultiRequests> element contains the following elements.

2970 <RequestReference> [one to many]

2971 Defines a request instance by reference to <Attributes> elements in the enclosing
2972 <Request> element. See section 5.51.

2973 5.51 Element <RequestReference>

2974 The <RequestReference> element defines an instance of a request in terms of references to
2975 <Attributes> elements. The semantics of this element are defined in [Multi]. Support for this element
2976 is optional.

```
2977 <xs:element name="RequestReference" type="xacml:RequestReference"/>
```

```

2978 <xs:complexType name="RequestReferenceType">
2979   <xs:sequence>
2980     <xs:element ref="xacml:AttributesReference" maxOccurs="unbounded"/>
2981   </xs:sequence>
2982 </xs:complexType>

```

2983 The <RequestReference> element contains the following elements.

2984 <AttributesReference> [one to many]

2985 A reference to an <Attributes> element in the enclosing <Request> element. See section
 2986 5.52.

2987 5.52 Element <AttributesReference>

2988 The <AttributesReference> element makes a reference to an <Attributes> element. The
 2989 meaning of this element is defined in [Multi]. Support for this element is optional.

```

2990 <xs:element name="AttributesReference" type="xacml:AttributesReference"/>
2991 <xs:complexType name="AttributesReferenceType">
2992   <xs:attribute name="ReferenceId" type="xs:IDREF" use="required" />
2993 </xs:complexType>

```

2994 The <AttributesReference> element contains the following attributes.

2995 ReferenceId [required]

2996 A reference to the xml:id attribute of an <Attributes> element in the enclosing <Request>
 2997 element.

2998 5.53 Element <Decision>

2999 The <Decision> element contains the result of *policy* evaluation.

```

3000 <xs:element name="Decision" type="xacml:DecisionType"/>
3001 <xs:simpleType name="DecisionType">
3002   <xs:restriction base="xs:string">
3003     <xs:enumeration value="Permit"/>
3004     <xs:enumeration value="Deny"/>
3005     <xs:enumeration value="Indeterminate"/>
3006     <xs:enumeration value="NotApplicable"/>
3007   </xs:restriction>
3008 </xs:simpleType>

```

3009 The <Decision> element is of DecisionType simple type.

3010 The values of the <Decision> element have the following meanings:

3011 "Permit": the requested **access** is permitted.

3012 "Deny": the requested **access** is denied.

3013 "Indeterminate": the **PDP** is unable to evaluate the requested **access**. Reasons for such inability
 3014 include: missing **attributes**, network errors while retrieving **policies**, division by zero during
 3015 **policy** evaluation, syntax errors in the **decision request** or in the **policy**, etc.

3016 "NotApplicable": the **PDP** does not have any **policy** that applies to this **decision request**.

3017 5.54 Element <Status>

3018 The <Status> element represents the status of the **authorization decision** result.

```

3019 <xs:element name="Status" type="xacml:StatusType"/>
3020 <xs:complexType name="StatusType">
3021   <xs:sequence>
3022     <xs:element ref="xacml:StatusCode"/>

```

3023
3024
3025
3026

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

3027 The <Status> element is of StatusType complex type.

3028 The <Status> element contains the following elements:

3029 <StatusCode> [Required]

3030 Status code.

3031 <StatusMessage> [Optional]

3032 A status message describing the status code.

3033 <StatusDetail> [Optional]

3034 Additional status information.

3035 **5.55 Element <StatusCode>**

3036 The <StatusCode> element contains a major status code value and an optional recursive series of
3037 minor status codes.

3038
3039
3040
3041
3042
3043
3044

```
<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>
```

3045 The <StatusCode> element is of StatusCodeType complex type.

3046 The <StatusCode> element contains the following attributes and elements:

3047 Value [Required]

3048 See Section B.8 for a list of values.

3049 <StatusCode> [Any Number]

3050 Minor status code. This status code qualifies its parent status code.

3051 **5.56 Element <StatusMessage>**

3052 The <StatusMessage> element is a free-form description of the status code.

3053

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

3054 The <StatusMessage> element is of xs:string type.

3055 **5.57 Element <StatusDetail>**

3056 The <StatusDetail> element qualifies the <Status> element with additional information.

3057
3058
3059
3060
3061
3062
3063

```
<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>
```

3064 The <StatusDetail> element is of StatusDetailType complex type.

3065 The <StatusDetail> element allows arbitrary XML content.

3066 Inclusion of a <StatusDetail> element is optional. However, if a **PDP** returns one of the following
3067 XACML-defined <StatusCode> values and includes a <StatusDetail> element, then the following
3068 rules apply.

3069 urn:oasis:names:tc:xacml:1.0:status:ok

3070 A **PDP** MUST NOT return a <StatusDetail> element in conjunction with the “ok” status value.

3071 urn:oasis:names:tc:xacml:1.0:status:missing-attribute

3072 A **PDP** MAY choose not to return any <StatusDetail> information or MAY choose to return a
3073 <StatusDetail> element containing one or more <MissingAttributeDetail> elements.

3074 urn:oasis:names:tc:xacml:1.0:status:syntax-error

3075 A **PDP** MUST NOT return a <StatusDetail> element in conjunction with the “syntax-error” status
3076 value. A syntax error may represent either a problem with the **policy** being used or with the request
3077 **context**. The **PDP** MAY return a <StatusMessage> describing the problem.

3078 urn:oasis:names:tc:xacml:1.0:status:processing-error

3079 A **PDP** MUST NOT return <StatusDetail> element in conjunction with the “processing-error” status
3080 value. This status code indicates an internal problem in the **PDP**. For security reasons, the **PDP** MAY
3081 choose to return no further information to the **PEP**. In the case of a divide-by-zero error or other
3082 computational error, the **PDP** MAY return a <StatusMessage> describing the nature of the error.

3083 5.58 Element <MissingAttributeDetail>

3084 The <MissingAttributeDetail> element conveys information about **attributes** required for **policy**
3085 evaluation that were missing from the request **context**.

```
3086 <xs:element name="MissingAttributeDetail"  
3087 type="xacml:MissingAttributeDetailType"/>  
3088 <xs:complexType name="MissingAttributeDetailType">  
3089 <xs:sequence>  
3090 <xs:element ref="xacml:AttributeValue" minOccurs="0"  
3091 maxOccurs="unbounded"/>  
3092 </xs:sequence>  
3093 <xs:attribute name="Category" type="xs:anyURI" use="required"/>  
3094 <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>  
3095 <xs:attribute name="DataType" type="xs:anyURI" use="required"/>  
3096 <xs:attribute name="Issuer" type="xs:string" use="optional"/>  
3097 </xs:complexType>
```

3098 The <MissingAttributeDetail> element is of MissingAttributeDetailType complex type.

3099 The <MissingAttributeDetail> element contains the following attributes and elements:

3100 <AttributeValue> [Optional]

3101 The required value of the missing **attribute**.

3102 Category [Required]

3103 The category identifier of the missing **attribute**.

3104 AttributeId [Required]

3105 The identifier of the missing **attribute**.

3106 DataType [Required]

3107 The data-type of the missing **attribute**.

3108 Issuer [Optional]

3109 This attribute, if supplied, SHALL specify the required Issuer of the missing **attribute**.

3110 If the **PDP** includes <AttributeValue> elements in the <MissingAttributeDetail> element, then
3111 this indicates the acceptable values for that **attribute**. If no <AttributeValue> elements are included,

3112 then this indicates the names of **attributes** that the **PDP** failed to resolve during its evaluation. The list of
3113 **attributes** may be partial or complete. There is no guarantee by the **PDP** that supplying the missing
3114 values or **attributes** will be sufficient to satisfy the **policy**.

3115 6 XPath 2.0 definitions

3116 The XPath 2.0 specification leaves a number of aspects of behavior implementation defined. This section
3117 defines how XPath 2.0 SHALL behave when hosted in XACML.

3118 <http://www.w3.org/TR/2007/REC-xpath20-20070123/#id-impl-defined-items> defines the following items:

- 3119 1. The version of Unicode that is used to construct expressions.
3120 XACML leaves this implementation defined. It is RECOMMENDED that the latest version is used.
- 3121 2. The statically-known collations.
3122 XACML leaves this implementation defined.
- 3123 3. The implicit timezone.
3124 XACML defined the implicit time zone as UTC.
- 3125 4. The circumstances in which warnings are raised, and the ways in which warnings are handled.
3126 XACML leaves this implementation defined.
- 3127 5. The method by which errors are reported to the external processing environment.
3128 An XPath error causes an XACML Indeterminate value in the element where the XPath error
3129 occurs. The StatusCode value SHALL be "urn:oasis:names:tc:xacml:1.0:status:processing-error".
3130 Implementations MAY provide additional details about the error in the response or by some other
3131 means.
- 3132 6. Whether the implementation is based on the rules of XML 1.0 or 1.1.
3133 XACML is based on XML 1.0.
- 3134 7. Whether the implementation supports the namespace axis.
3135 XACML leaves this implementation defined. It is RECOMMENDED that users of XACML do not
3136 make use of the namespace axis.
- 3137 8. Any static typing extensions supported by the implementation, if the Static Typing Feature is
3138 supported.
3139 XACML leaves this implementation defined.

3140

3141 <http://www.w3.org/TR/2007/REC-xpath-datamodel-20070123/#implementation-defined> defines the
3142 following items:

- 3143 1. Support for additional user-defined or implementation-defined types is implementation-defined.
3144 It is RECOMMENDED that implementations of XACML do not define any additional types and it is
3145 RECOMMENDED that users of XACML do not make user of any additional types.
- 3146 2. Some typed values in the data model are undefined. Attempting to access an undefined property
3147 is always an error. Behavior in these cases is implementation-defined and the host language is
3148 responsible for determining the result.
3149 An XPath error causes an XACML Indeterminate value in the element where the XPath error
3150 occurs. The StatusCode value SHALL be "urn:oasis:names:tc:xacml:1.0:status:processing-error".
3151 Implementations MAY provide additional details about the error in the response or by some other
3152 means.

3153

3154 <http://www.w3.org/TR/2007/REC-xpath-functions-20070123/#impl-def> defines the following items:

- 3155 1. The destination of the trace output is implementation-defined.
3156 XACML leaves this implementation defined.
- 3157 2. For xs:integer operations, implementations that support limited-precision integer operations must
3158 either raise an error [err:FOAR0002] or provide an implementation-defined mechanism that
3159 allows users to choose between raising an error and returning a result that is modulo the largest
3160 representable integer value.
3161 XACML leaves this implementation defined. If an implementation chooses to raise an error, the

- 3162 StatusCode value SHALL be “urn:oasis:names:tc:xacml:1.0:status:processing-error”.
- 3163 Implementations MAY provide additional details about the error in the response or by some other
- 3164 means.
- 3165 3. For xs:decimal values the number of digits of precision returned by the numeric operators is
- 3166 implementation-defined.
- 3167 XACML leaves this implementation defined.
- 3168 4. If the number of digits in the result of a numeric operation exceeds the number of digits that the
- 3169 implementation supports, the result is truncated or rounded in an implementation-defined manner.
- 3170 XACML leaves this implementation defined.
- 3171 5. It is implementation-defined which version of Unicode is supported.
- 3172 XACML leaves this implementation defined. It is RECOMMENDED that the latest version is used.
- 3173 6. For fn:normalize-unicode, conforming implementations must support normalization form "NFC"
- 3174 and may support normalization forms "NFD", "NFKC", "NFKD", "FULLY-NORMALIZED". They
- 3175 may also support other normalization forms with implementation-defined semantics.
- 3176 XACML leaves this implementation defined.
- 3177 7. The ability to decompose strings into collation units suitable for substring matching is an
- 3178 implementation-defined property of a collation.
- 3179 XACML leaves this implementation defined.
- 3180 8. All minimally conforming processors must support year values with a minimum of 4 digits (i.e.,
- 3181 YYYY) and a minimum fractional second precision of 1 millisecond or three digits (i.e., s.sss).
- 3182 However, conforming processors may set larger implementation-defined limits on the maximum
- 3183 number of digits they support in these two situations.
- 3184 XACML leaves this implementation defined, and it is RECOMMENDED that users of XACML do
- 3185 not expect greater limits and precision.
- 3186 9. The result of casting a string to xs:decimal, when the resulting value is not too large or too small
- 3187 but nevertheless has too many decimal digits to be accurately represented, is implementation-
- 3188 defined.
- 3189 XACML leaves this implementation defined.
- 3190 10. Various aspects of the processing provided by fn:doc are implementation-defined.
- 3191 Implementations may provide external configuration options that allow any aspect of the
- 3192 processing to be controlled by the user.
- 3193 XACML leaves this implementation defined.
- 3194 11. The manner in which implementations provide options to weaken the stable characteristic of
- 3195 fn:collection and fn:doc are implementation-defined.
- 3196 XACML leaves this implementation defined.

3197 7 Functional requirements

3198 This section specifies certain functional requirements that are not directly associated with the production
3199 or consumption of a particular XACML element.

3200 Note that in each case an implementation is conformant as long as it produces the same result as is
3201 specified here, regardless of how and in what order the implementation behaves internally.

3202 7.1 Unicode issues

3203 7.1.1 Normalization

3204 In Unicode, some equivalent characters can be represented by more than one different Unicode
3205 character sequence. See [CMF]. The process of converting Unicode strings into equivalent character
3206 sequences is called "normalization" [UAX15]. Some operations, such as string comparison, are sensitive
3207 to normalization. An operation is normalization-sensitive if its output(s) are different depending on the
3208 state of normalization of the input(s); if the output(s) are textual, they are deemed different only if they
3209 would remain different were they to be normalized.

3210 For more information on normalization see [CM].

3211 An XACML implementation MUST behave as if each normalization-sensitive operation normalizes input
3212 strings into Unicode Normalization Form C ("NFC"). An implementation MAY use some other form of
3213 internal processing (such as using a non-Unicode, "legacy" character encoding) as long as the externally
3214 visible results are identical to this specification.

3215 7.1.2 Version of Unicode

3216 The version of Unicode used by XACML is implementation defined. It is RECOMMENDED that the latest
3217 version is used. Also note security issues in section 9.3.

3218 7.2 Policy enforcement point

3219 This section describes the requirements for the *PEP*.

3220 An application functions in the role of the *PEP* if it guards *access* to a set of *resources* and asks the
3221 *PDP* for an *authorization decision*. The *PEP* MUST abide by the *authorization decision* as described
3222 in one of the following sub-sections

3223 In any case any *advice* in the *decision* may be safely ignored by the *PEP*.

3224 7.2.1 Base PEP

3225 If the *decision* is "Permit", then the *PEP* SHALL permit *access*. If *obligations* accompany the *decision*,
3226 then the *PEP* SHALL permit *access* only if it understands and it can and will discharge those
3227 *obligations*.

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

3230 If the *decision* is "Not Applicable", then the *PEP*'s behavior is undefined.

3231 If the *decision* is "Indeterminate", then the *PEP*'s behavior is undefined.

3232 7.2.2 Deny-biased PEP

3233 If the *decision* is "Permit", then the *PEP* SHALL permit *access*. If *obligations* accompany the *decision*,
3234 then the *PEP* SHALL permit *access* only if it understands and it can and will discharge those
3235 *obligations*.

3236 All other **decisions** SHALL result in the denial of **access**.
3237 Note: other actions, e.g. consultation of additional **PDPs**, reformulation/resubmission of
3238 the **decision request**, etc., are not prohibited.

3239 7.2.3 Permit-biased PEP

3240 If the **decision** is "Deny", then the **PEP** SHALL deny **access**. If **obligations** accompany the **decision**,
3241 then the **PEP** shall deny **access** only if it understands, and it can and will discharge those **obligations**.

3242 All other **decisions** SHALL result in the permission of **access**.

3243 Note: other actions, e.g. consultation of additional **PDPs**, reformulation/resubmission of
3244 the **decision request**, etc., are not prohibited.

3245 7.3 Attribute evaluation

3246 **Attributes** are represented in the request **context** by the **context handler**, regardless of whether or not
3247 they appeared in the original **decision request**, and are referred to in the **policy** by attribute designators
3248 and attribute selectors. A **named attribute** is the term used for the criteria that the specific attribute
3249 designators use to refer to particular **attributes** in the <Attributes> elements of the request **context**.

3250 7.3.1 Structured attributes

3251 <AttributeValue> elements MAY contain an instance of a structured XML data-type, for example
3252 <ds:KeyInfo>. XACML 3.0 supports several ways for comparing the contents of such elements.

3253 1. In some cases, such elements MAY be compared using one of the XACML string functions, such
3254 as "string-regexp-match", described below. This requires that the element be given the data-type
3255 "http://www.w3.org/2001/XMLSchema#string". For example, a structured data-type that is
3256 actually a ds:KeyInfo/KeyName would appear in the **Context** as:

```
3257 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">  
3258 <ds:KeyName>jhibbert-key</ds:KeyName>  
3259 </AttributeValue>
```

3260 In general, this method will not be adequate unless the structured data-type is quite simple.

3261 2. The structured **attribute** MAY be made available in the <Content> element of the appropriate
3262 **attribute** category and an <AttributeSelector> element MAY be used to select the contents
3263 of a leaf sub-element of the structured data-type by means of an XPath expression. That value
3264 MAY then be compared using one of the supported XACML functions appropriate for its primitive
3265 data-type. This method requires support by the **PDP** for the optional XPath expressions feature.

3266 3. The structured **attribute** MAY be made available in the <Content> element of the appropriate
3267 **attribute** category and an <AttributeSelector> element MAY be used to select any node in
3268 the structured data-type by means of an XPath expression. This node MAY then be compared
3269 using one of the XPath-based functions described in Section A.3.15. This method requires
3270 support by the **PDP** for the optional XPath expressions and XPath functions features.

3271 See also Section 7.3.

3272 7.3.2 Attribute bags

3273 XACML defines implicit collections of its data-types. XACML refers to a collection of values that are of a
3274 single data-type as a **bag**. **Bags** of data-types are needed because selections of nodes from an XML
3275 **resource** or XACML request **context** may return more than one value.

3276 The <AttributeSelector> element uses an XPath expression to specify the selection of data from
3277 free form XML. The result of an XPath expression is termed a node-set, which contains all the nodes
3278 from the XML content that match the **predicate** in the XPath expression. Based on the various indexing
3279 functions provided in the XPath specification, it SHALL be implied that a resultant node-set is the

3280 collection of the matching nodes. XACML also defines the <AttributeDesignator> element to have
3281 the same matching methodology for **attributes** in the XACML request **context**.

3282 The values in a **bag** are not ordered, and some of the values may be duplicates. There SHALL be no
3283 notion of a **bag** containing **bags**, or a **bag** containing values of differing types; i.e., a **bag** in XACML
3284 SHALL contain only values that are of the same data-type.

3285 7.3.3 Multivalued attributes

3286 If a single <Attribute> element in a request **context** contains multiple <AttributeValue> child
3287 elements, then the **bag** of values resulting from evaluation of the <Attribute> element MUST be
3288 identical to the **bag** of values that results from evaluating a **context** in which each <AttributeValue>
3289 element appears in a separate <Attribute> element, each carrying identical meta-data.

3290 7.3.4 Attribute Matching

3291 A **named attribute** includes specific criteria with which to match **attributes** in the **context**. An **attribute**
3292 specifies a *Category*, *AttributeId* and *DataType*, and a **named attribute** also specifies the
3293 *Issuer*. A **named attribute** SHALL match an **attribute** if the values of their respective *Category*,
3294 *AttributeId*, *DataType* and optional *Issuer* attributes match. The *Category* of the **named**
3295 **attribute** MUST match, by **identifier equality**, the *Category* of the corresponding **context attribute**.
3296 The *AttributeId* of the **named attribute** MUST match, by **identifier equality**, the *AttributeId* of
3297 the corresponding **context attribute**. The *DataType* of the **named attribute** MUST match, by **identifier**
3298 **equality**, the *DataType* of the corresponding **context attribute**. If *Issuer* is supplied in the **named**
3299 **attribute**, then it MUST match, using the urn:oasis:names:tc:xacml:1.0:function:string-equal function, the
3300 *Issuer* of the corresponding **context attribute**. If *Issuer* is not supplied in the **named attribute**, then
3301 the matching of the **context attribute** to the **named attribute** SHALL be governed by *AttributeId* and
3302 *DataType* alone, regardless of the presence, absence, or actual value of *Issuer* in the corresponding
3303 **context attribute**. In the case of an attribute selector, the matching of the **attribute** to the **named**
3304 **attribute** SHALL be governed by the XPath expression and *DataType*.

3305 7.3.5 Attribute Retrieval

3306 The **PDP** SHALL request the values of **attributes** in the request **context** from the **context handler**. The
3307 **context handler** MAY also add **attributes** to the request **context** without the **PDP** requesting them. The
3308 **PDP** SHALL reference the **attributes** as if they were in a physical request **context** document, but the
3309 **context handler** is responsible for obtaining and supplying the requested values by whatever means it
3310 deems appropriate, including by retrieving them from one or more Policy Information Points. The **context**
3311 **handler** SHALL return the values of **attributes** that match the attribute designator or attribute selector
3312 and form them into a **bag** of values with the specified data-type. If no **attributes** from the request
3313 **context** match, then the **attribute** SHALL be considered missing. If the **attribute** is missing, then
3314 *MustBePresent* governs whether the attribute designator or attribute selector returns an empty **bag** or
3315 an “Indeterminate” result. If *MustBePresent* is “False” (default value), then a missing **attribute** SHALL
3316 result in an empty **bag**. If *MustBePresent* is “True”, then a missing **attribute** SHALL result in
3317 “Indeterminate”. This “Indeterminate” result SHALL be handled in accordance with the specification of the
3318 encompassing expressions, **rules**, **policies** and **policy sets**. If the result is “Indeterminate”, then the
3319 *AttributeId*, *DataType* and *Issuer* of the **attribute** MAY be listed in the **authorization decision** as
3320 described in Section 7.17. However, a **PDP** MAY choose not to return such information for security
3321 reasons.

3322 Regardless of any dynamic modifications of the request **context** during policy evaluation, the **PDP**
3323 SHALL behave as if each bag of **attribute** values is fully populated in the **context** before it is first tested,
3324 and is thereafter immutable during evaluation. (That is, every subsequent test of that **attribute** shall use
3325 the same bag of values that was initially tested.)

3326 **7.3.6 Environment Attributes**

3327 Standard *environment attributes* are listed in Section B.7. If a value for one of these *attributes* is
3328 supplied in the *decision request*, then the *context handler* SHALL use that value. Otherwise, the
3329 *context handler* SHALL supply a value. In the case of date and time *attributes*, the supplied value
3330 SHALL have the semantics of the "date and time that apply to the *decision request*".

3331 **7.3.7 AttributeSelector evaluation**

3332 An <AttributeSelector> element will be evaluated according to the following processing model.

3333

3334 NOTE: It is not necessary for an implementation to actually follow these steps. It is only
3335 necessary to produce results identical to those that would be produced by following these
3336 steps.

- 3337 1. If the *attributes* category given by the `Category` attribute is not found or does not have a
3338 <Content> child element, then the return value is either "Indeterminate" or an empty *bag* as
3339 determined by the `MustBePresent` attribute; otherwise, construct an XML data structure
3340 suitable for xpath processing from the <Content> element in the *attributes* category given by
3341 the `Category` attribute. The data structure shall be constructed so that the document node of
3342 this structure contains a single document element which corresponds to the single child element
3343 of the <Content> element. The constructed data structure shall be equivalent to one that would
3344 result from parsing a stand-alone XML document consisting of the contents of the <Content>
3345 element (including any comment and processing-instruction markup). Namespace declarations
3346 which are not "visibly utilized", as defined by [exc-c14n], MAY not be present and MUST NOT be
3347 utilized by the XPath expression in step 3. The data structure must meet the requirements of the
3348 applicable xpath version.
- 3349 2. Select a context node for xpath processing from this data structure. If there is a
3350 `ContextSelectorId` attribute, the context node shall be the node selected by applying the
3351 XPath expression given in the *attribute* value of the designated *attribute* (in the *attributes*
3352 category given by the <AttributeSelector> `Category` attribute). It shall be an error if this
3353 evaluation returns no node or more than one node, in which case the return value MUST be an
3354 "Indeterminate" with a status code "urn:oasis:names:tc:xacml:1.0:status:syntax-error". If there is
3355 no `ContextSelectorId`, the document node of the data structure shall be the context node.
- 3356 3. Evaluate the XPath expression given in the `Path` attribute against the xml data structure, using
3357 the context node selected in the previous step. It shall be an error if this evaluation returns
3358 anything other than a sequence of nodes (possibly empty), in which case the
3359 <AttributeSelector> MUST return "Indeterminate" with a status code
3360 "urn:oasis:names:tc:xacml:1.0:status:syntax-error". If the evaluation returns an empty sequence
3361 of nodes, then the return value is either "Indeterminate" or an empty *bag* as determined by the
3362 `MustBePresent` attribute.
- 3363 4. If the data type is a primitive data type, convert the text value of each selected node to the
3364 desired data type, as specified in the `DataType` attribute. Each value shall be constructed using
3365 the appropriate constructor function from [XF] Section 5 listed below, corresponding to the
3366 specified data type.

- 3367
3368 `xs:string()`
3369 `xs:boolean()`
3370 `xs:integer()`
3371 `xs:double()`
3372 `xs:dateTime()`
3373 `xs:date()`
3374 `xs:time()`
3375 `xs:hexBinary()`
3376 `xs:base64Binary()`

3377 xs:anyURI()
3378 xs:yearMonthDuration()
3379 xs:dayTimeDuration()
3380
3381 If the `DataType` is not one of the primitive types listed above, then the return values shall be
3382 constructed from the nodeset in a manner specified by the particular `DataType` extension
3383 specification. If the data type extension does not specify an appropriate constructor function, then
3384 the `<AttributeSelector>` MUST return "Indeterminate" with a status code
3385 "urn:oasis:names:tc:xacml:1.0:status:syntax-error".
3386
3387 If an error occurs when converting the values returned by the XPath expression to the specified
3388 `DataType`, then the result of the `<AttributeSelector>` MUST be "Indeterminate", with a
3389 status code "urn:oasis:names:tc:xacml:1.0:status:processing-error"

3390 7.4 Expression evaluation

3391 XACML specifies expressions in terms of the elements listed below, of which the `<Apply>` and
3392 `<Condition>` elements recursively compose greater expressions. Valid expressions SHALL be type
3393 correct, which means that the types of each of the elements contained within `<Apply>` elements SHALL
3394 agree with the respective argument types of the function that is named by the `FunctionId` attribute.
3395 The resultant type of the `<Apply>` element SHALL be the resultant type of the function, which MAY be
3396 narrowed to a primitive data-type, or a **bag** of a primitive data-type, by type-unification. XACML defines
3397 an evaluation result of "Indeterminate", which is said to be the result of an invalid expression, or an
3398 operational error occurring during the evaluation of the expression.

3399 XACML defines these elements to be in the substitution group of the `<Expression>` element:

- 3400 • `<xacml:AttributeValue>`
- 3401 • `<xacml:AttributeDesignator>`
- 3402 • `<xacml:AttributeSelector>`
- 3403 • `<xacml:Apply>`
- 3404 • `<xacml:Function>`
- 3405 • `<xacml:VariableReference>`

3406 7.5 Arithmetic evaluation

3407 IEEE 754 [IEEE754] specifies how to evaluate arithmetic functions in a context, which specifies defaults
3408 for precision, rounding, etc. XACML SHALL use this specification for the evaluation of all integer and
3409 double functions relying on the Extended Default Context, enhanced with double precision:

- 3410 flags - all set to 0
- 3411 trap-enablers - all set to 0 (IEEE 854 §7) with the exception of the "division-by-zero" trap enabler,
3412 which SHALL be set to 1
- 3413 precision - is set to the designated double precision
- 3414 rounding - is set to round-half-even (IEEE 854 §4.1)

3415 7.6 Match evaluation

3416 The **attribute** matching element `<Match>` appears in the `<Target>` element of **rules**, **policies** and
3417 **policy sets**.

3418 This element represents a Boolean expression over **attributes** of the request **context**. A matching
3419 element contains a `MatchId` attribute that specifies the function to be used in performing the match
3420 evaluation, an `<AttributeValue>` and an `<AttributeDesignator>` or `<AttributeSelector>`
3421 element that specifies the **attribute** in the **context** that is to be matched against the specified value.

3422 The `MatchId` attribute SHALL specify a function that takes two arguments, returning a result type of
3423 "http://www.w3.org/2001/XMLSchema#boolean". The **attribute** value specified in the matching element
3424 SHALL be supplied to the `MatchId` function as its first argument. An element of the **bag** returned by the
3425 `<AttributeDesignator>` or `<AttributeSelector>` element SHALL be supplied to the `MatchId`
3426 function as its second argument, as explained below. The `DataType` of the `<AttributeValue>`
3427 SHALL match the data-type of the first argument expected by the `MatchId` function. The `DataType` of
3428 the `<AttributeDesignator>` or `<AttributeSelector>` element SHALL match the data-type of the
3429 second argument expected by the `MatchId` function.

3430 In addition, functions that are strictly within an extension to XACML MAY appear as a value for the
3431 `MatchId` attribute, and those functions MAY use data-types that are also extensions, so long as the
3432 extension function returns a Boolean result and takes two single base types as its inputs. The function
3433 used as the value for the `MatchId` attribute SHOULD be easily indexable. Use of non-indexable or
3434 complex functions may prevent efficient evaluation of **decision requests**.

3435 The evaluation semantics for a matching element is as follows. If an operational error were to occur while
3436 evaluating the `<AttributeDesignator>` or `<AttributeSelector>` element, then the result of the
3437 entire expression SHALL be "Indeterminate". If the `<AttributeDesignator>` or
3438 `<AttributeSelector>` element were to evaluate to an empty **bag**, then the result of the expression
3439 SHALL be "False". Otherwise, the `MatchId` function SHALL be applied between the
3440 `<AttributeValue>` and each element of the **bag** returned from the `<AttributeDesignator>` or
3441 `<AttributeSelector>` element. If at least one of those function applications were to evaluate to
3442 "True", then the result of the entire expression SHALL be "True". Otherwise, if at least one of the function
3443 applications results in "Indeterminate", then the result SHALL be "Indeterminate". Finally, if all function
3444 applications evaluate to "False", then the result of the entire expression SHALL be "False".

3445 It is also possible to express the semantics of a **target** matching element in a **condition**. For instance,
3446 the **target** match expression that compares a "**subject-name**" starting with the name "John" can be
3447 expressed as follows:

```
3448 <Match
3449 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
3450   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
3451     John.*
3452   </AttributeValue>
3453   <AttributeDesignator
3454     Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
3455 subject"
3456     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
3457     DataType="http://www.w3.org/2001/XMLSchema#string"/>
3458 </Match>
```

3459 Alternatively, the same match semantics can be expressed as an `<Apply>` element in a **condition** by
3460 using the "urn:oasis:names:tc:xacml:3.0:function:any-of" function, as follows:

```
3461 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">
3462   <Function
3463     FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match"/>
3464   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
3465     John.*
3466   </AttributeValue>
3467   <AttributeDesignator
3468     Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
3469 subject"
3470     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
3471     DataType="http://www.w3.org/2001/XMLSchema#string"/>
3472 </Apply>
```

3473 **7.7 Target evaluation**

3474 An empty **target** matches any request. Otherwise the **target** value SHALL be "Match" if all the AnyOf
 3475 specified in the **target** match values in the request **context**. Otherwise, if any one of the AnyOf specified
 3476 in the **target** is "No Match", then the **target** SHALL be "No Match". Otherwise, the **target** SHALL be
 3477 "Indeterminate". The **target** match table is shown in Table 1.

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

3478 *Table 1 Target match table*

3479 The AnyOf SHALL match values in the request **context** if at least one of their <AllOf> elements
 3480 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"

3481 *Table 2 AnyOf match table*

3482 An AllOf SHALL match a value in the request **context** if the value of all its <Match> elements is "True".

3483 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"

3484 *Table 3 AllOf match table*

3485 **7.8 VariableReference Evaluation**

3486 The <VariableReference> element references a single <VariableDefinition> element contained
 3487 within the same <Policy> element. A <VariableReference> that does not reference a particular
 3488 <VariableDefinition> element within the encompassing <Policy> element is called an undefined
 3489 reference. **Policies** with undefined references are invalid.

3490 In any place where a <VariableReference> occurs, it has the effect as if the text of the
 3491 <Expression> element defined in the <VariableDefinition> element replaces the
 3492 <VariableReference> element. Any evaluation scheme that preserves this semantic is acceptable.
 3493 For instance, the expression in the <VariableDefinition> element may be evaluated to a particular
 3494 value and cached for multiple references without consequence. (I.e. the value of an <Expression>
 3495 element remains the same for the entire **policy** evaluation.) This characteristic is one of the benefits of
 3496 XACML being a declarative language.

3497 A variable reference containing circular references is invalid. The PDP MUST detect circular references
 3498 either at policy loading time or during runtime evaluation. If the PDP detects a circular reference during

3499 runtime the variable reference evaluates to "Indeterminate" with status code
 3500 urn:oasis:names:tc:xacml:1.0:status:processing-error.

3501 7.9 Condition evaluation

3502 The **condition** value SHALL be "True" if the <Condition> element is absent, or if it evaluates to "True".
 3503 Its value SHALL be "False" if the <Condition> element evaluates to "False". The **condition** value
 3504 SHALL be "Indeterminate", if the expression contained in the <Condition> element evaluates to
 3505 "Indeterminate."

3506 7.10 Extended Indeterminate

3507 Some **combining algorithms** are defined in terms of an extended set of "Indeterminate" values. The
 3508 extended set associated with the "Indeterminate" contains the potential effect values which could have
 3509 occurred if there would not have been an error causing the "Indeterminate". The possible extended set
 3510 "Indeterminate" values are

- 3511 • "Indeterminate{D}": an "Indeterminate" from a **policy** or **rule** which could have evaluated to "Deny",
 3512 but not "Permit"
- 3513 • "Indeterminate{P}": an "Indeterminate" from a **policy** or **rule** which could have evaluated to "Permit",
 3514 but not "Deny"
- 3515 • "Indeterminate{DP}": an "Indeterminate" from a **policy** or **rule** which could have evaluated to "Deny"
 3516 or "Permit".

3517 The **combining algorithms** which are defined in terms of the extended "Indeterminate" make use of the
 3518 additional information to allow for better treatment of errors in the algorithms.

3519 The final decision returned by a **PDP** cannot be an extended Indeterminate. Any such decision at the top
 3520 level **policy** or **policy set** is returned as a plain Indeterminate in the response from the **PDP**.

3521 The tables in the following four sections define how extended "Indeterminate" values are produced during
 3522 **Rule**, **Policy** and **PolicySet** evaluation.

3523 7.11 Rule evaluation

3524 A **rule** has a value that can be calculated by evaluating its contents. **Rule** evaluation involves separate
 3525 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"	Effect
"Match" or no target	"False"	"NotApplicable"
"Match" or no target	"Indeterminate"	"Indeterminate{P}" if the Effect is Permit, or "Indeterminate{D}" if the Effect is Deny
"No-match"	Don't care	"NotApplicable"
"Indeterminate"	Don't care	"Indeterminate{P}" if the Effect is Permit, or "Indeterminate{D}" if the Effect is Deny

3526 Table 4 Rule truth table.

3527 7.12 Policy evaluation

3528 The value of a **policy** SHALL be determined only by its contents, considered in relation to the contents of
 3529 the request **context**. A **policy's** value SHALL be determined by evaluation of the **policy's target** and,
 3530 according to the specified **rule-combining algorithm, rules**,.

3531 The **policy** truth table is shown in Table 5.

Target	Rule values	Policy Value
“Match”	Don’t care	Specified by the rule-combining algorithm
“No-match”	Don’t care	“NotApplicable”
“Indeterminate”	See Table 7	See Table 7

3532 Table 5 Policy truth table

3533 Note that none of the **rule-combining algorithms** defined by XACML 3.0 take parameters. However,
 3534 non-standard combining algorithms MAY take parameters. In such a case, the values of these
 3535 parameters associated with the **rules**, MUST be taken into account when evaluating the **policy**. The
 3536 parameters and their types should be defined in the specification of the combining algorithm. If the
 3537 implementation supports combiner parameters and if combiner parameters are present in a **policy**, then
 3538 the parameter values MUST be supplied to the combining algorithm implementation.

3539 7.13 Policy Set evaluation

3540 The value of a **policy set** SHALL be determined by its contents, considered in relation to the contents of
 3541 the request **context**. A **policy set**'s value SHALL be determined by evaluation of the **policy set**'s **target**,
 3542 and, according to the specified **policy-combining algorithm**, **policies** and **policy sets**,

3543 The **policy set** truth table is shown in Table 6.

Target	Policy values	Policy set Value
“Match”	Don’t care	Specified by the policy-combining algorithm
“No-match”	Don’t care	“NotApplicable”
“Indeterminate”	See Table 7	See Table 7

3544 Table 6 Policy set truth table

3545 Note that none of the **policy-combining algorithms** defined by XACML 3.0 take parameters. However,
 3546 non-standard combining algorithms MAY take parameters. In such a case, the values of these
 3547 parameters associated with the **policies**, MUST be taken into account when evaluating the **policy set**.
 3548 The parameters and their types should be defined in the specification of the combining algorithm. If the
 3549 implementation supports combiner parameters and if combiner parameters are present in a **policy**, then
 3550 the parameter values MUST be supplied to the combining algorithm implementation.

3551 7.14 Policy and Policy set value for Indeterminate Target

3552 If the **target** of a **policy** or **policy set** evaluates to “Indeterminate”, the value of the **policy** or **policy set**
 3553 as a whole is determined by the value of the **combining algorithm** according to Table 7.

Combining algorithm Value	Policy set or policy Value
“NotApplicable”	“NotApplicable”
“Permit”	“Indeterminate{P}”
“Deny”	“Indeterminate{D}”
“Indeterminate”	“Indeterminate{DP}”
“Indeterminate{DP}”	“Indeterminate{DP}”
“Indeterminate{P}”	“Indeterminate{P}”

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

3554 Table 7 The value of a **policy** or **policy set** when the target is "Indeterminate".

3555 7.15 PolicySetIdReference and PolicyIdReference evaluation

3556 A policy set id reference or a policy id reference is evaluated by resolving the reference and evaluating
3557 the referenced policy set or policy.

3558 If resolving the reference fails, the reference evaluates to "Indeterminate" with status code
3559 urn:oasis:names:tc:xacml:1.0:status:processing-error.

3560 A policy set id reference or a policy id reference containing circular references is invalid. The PDP MUST
3561 detect circular references either at policy loading time or during runtime evaluation. If the PDP detects a
3562 circular reference during runtime the reference evaluates to "Indeterminate" with status code
3563 urn:oasis:names:tc:xacml:1.0:status:processing-error.

3564 7.16 Hierarchical resources

3565 It is often the case that a **resource** is organized as a hierarchy (e.g. file system, XML document). XACML
3566 provides several optional mechanisms for supporting hierarchical **resources**. These are described in the
3567 XACML Profile for Hierarchical Resources [**Hier**] and in the XACML Profile for Requests for Multiple
3568 Resources [**Multi**].

3569 7.17 Authorization decision

3570 In relation to a particular **decision request**, the **PDP** is defined by a **policy-combining algorithm** and a
3571 set of **policies** and/or **policy sets**. The **PDP** SHALL return a response **context** as if it had evaluated a
3572 single **policy set** consisting of this **policy-combining algorithm** and the set of **policies** and/or **policy**
3573 **sets**.

3574 The **PDP** MUST evaluate the **policy set** as specified in Sections 5 and 7. The **PDP** MUST return a
3575 response **context**, with one <Decision> element of value "Permit", "Deny", "Indeterminate" or
3576 "NotApplicable".

3577 If the **PDP** cannot make a **decision**, then an "Indeterminate" <Decision> element SHALL be returned.

3578 7.18 Obligations and advice

3579 A **rule**, **policy**, or **policy set** may contain one or more **obligation** or **advice** expressions. When such a
3580 **rule**, **policy**, or **policy set** is evaluated, the **obligation** or **advice** expression SHALL be evaluated to an
3581 **obligation** or **advice** respectively, which SHALL be passed up to the next level of evaluation (the
3582 enclosing or referencing **policy**, **policy set**, or **authorization decision**) only if the result of the **rule**,
3583 **policy**, or **policy set** being evaluated matches the value of the FulfillOn attribute of the **obligation** or
3584 the AppliesTo attribute of the **advice**. If any of the **attribute** assignment expressions in an **obligation**
3585 or **advice** expression with a matching FulfillOn or AppliesTo attribute evaluates to "Indeterminate",
3586 then the whole **rule**, **policy**, or **policy set** SHALL be "Indeterminate". If the FulfillOn or AppliesTo
3587 attribute does not match the result of the combining algorithm or the **rule** evaluation, then any
3588 indeterminate in an **obligation** or **advice** expression has no effect.

3589 As a consequence of this procedure, no **obligations** or **advice** SHALL be returned to the **PEP** if the **rule**,
3590 **policies**, or **policy sets** from which they are drawn are not evaluated, or if their evaluated result is
3591 "Indeterminate" or "NotApplicable", or if the **decision** resulting from evaluating the **rule**, **policy**, or **policy**
3592 **set** does not match the **decision** resulting from evaluating an enclosing **policy set**.

3593 If the **PDP**'s evaluation is viewed as a tree of **rules**, **policy sets** and **policies**, each of which returns
3594 "Permit" or "Deny", then the set of **obligations** and **advice** returned by the **PDP** to the **PEP** will include
3595 only the **obligations** and **advice** associated with those paths where the result at each level of evaluation
3596 is the same as the result being returned by the **PDP**. In situations where any lack of determinism is
3597 unacceptable, a deterministic combining algorithm, such as ordered-deny-overrides, should be used.

3598 Also see Section 7.2.

3599 7.19 Exception handling

3600 XACML specifies behavior for the **PDP** in the following situations.

3601 7.19.1 Unsupported functionality

3602 If the **PDP** attempts to evaluate a **policy set** or **policy** that contains an optional element type or function
3603 that the **PDP** does not support, then the **PDP** SHALL return a <Decision> value of "Indeterminate". If a
3604 <StatusCode> element is also returned, then its value SHALL be
3605 "urn:oasis:names:tc:xacml:1.0:status:syntax-error" in the case of an unsupported element type, and
3606 "urn:oasis:names:tc:xacml:1.0:status:processing-error" in the case of an unsupported function.

3607 7.19.2 Syntax and type errors

3608 If a **policy** that contains invalid syntax is evaluated by the XACML **PDP** at the time a **decision request** is
3609 received, then the result of that **policy** SHALL be "Indeterminate" with a `StatusCode` value of
3610 "urn:oasis:names:tc:xacml:1.0:status:syntax-error".

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

3614 7.19.3 Missing attributes

3615 The absence of matching **attributes** in the request **context** for any of the attribute designators attribute or
3616 selectors that are found in the **policy** will result in an enclosing <AllOf> element to return a value of
3617 "Indeterminate", if the designator or selector has the `MustBePresent` XML attribute set to true, as
3618 described in Sections 5.29 and 5.30 and may result in a <Decision> element containing the
3619 "Indeterminate" value. If, in this case a status code is supplied, then the value

3620 "urn:oasis:names:tc:xacml:1.0:status:missing-attribute"

3621 SHALL be used, to indicate that more information is needed in order for a definitive **decision** to be
3622 rendered. In this case, the <Status> element MAY list the names and data-types of any **attributes** that
3623 are needed by the **PDP** to refine its **decision** (see Section 5.58). A **PEP** MAY resubmit a refined request
3624 **context** in response to a <Decision> element contents of "Indeterminate" with a status code of

3625 "urn:oasis:names:tc:xacml:1.0:status:missing-attribute"

3626 by adding **attribute** values for the **attribute** names that were listed in the previous response. When the
3627 **PDP** returns a <Decision> element contents of "Indeterminate", with a status code of

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

3629 it MUST NOT list the names and data-types of any **attribute** for which values were supplied in the original
3630 request. Note, this requirement forces the **PDP** to eventually return an **authorization decision** of
3631 "Permit", "Deny", or "Indeterminate" with some other status code, in response to successively-refined
3632 requests.

3633 7.20 Identifier equality

3634 XACML makes use of URIs and strings as identifiers. When such identifiers are compared for equality,
3635 the comparison MUST be done so that the identifiers are equal if they have the same length and the
3636 characters in the two identifiers are equal codepoint by codepoint.

3637 The following is a list of the identifiers which MUST use this definition of equality.

3638 The content of the element <XPathVersion>.

3639 The XML attribute `Value` in the element <StatusCode>.

- 3640 The XML attributes `Category`, `AttributeId`, `DataType` and `Issuer` in the element
3641 `<MissingAttributeDetail>`.
- 3642 The XML attribute `Category` in the element `<Attributes>`.
- 3643 The XML attributes `AttributeId` and `Issuer` in the element `<Attribute>`.
- 3644 The XML attribute `ObligationId` in the element `<Obligation>`.
- 3645 The XML attribute `AdviceId` in the element `<Advice>`.
- 3646 The XML attributes `AttributeId` and `Category` in the element `<AttributeAssignment>`.
- 3647 The XML attribute `ObligationId` in the element `<ObligationExpression>`.
- 3648 The XML attribute `AdviceId` in the element `<AdviceExpression>`.
- 3649 The XML attributes `AttributeId`, `Category` and `Issuer` in the element
3650 `<AttributeAssignmentExpression>`.
- 3651 The XML attributes `PolicySetId` and `PolicyCombiningAlgId` in the element `<PolicySet>`.
- 3652 The XML attribute `ParameterName` in the element `<CombinerParameter>`.
- 3653 The XML attribute `RuleIdRef` in the element `<RuleCombinerParameters>`.
- 3654 The XML attribute `PolicyIdRef` in the element `<PolicyCombinerParameters>`.
- 3655 The XML attribute `PolicySetIdRef` in the element `<PolicySetCombinerParameters>`.
- 3656 The anyURI in the content of the complex type `IdReferenceType`.
- 3657 The XML attributes `PolicyId` and `RuleCombiningAlgId` in the element `<Policy>`.
- 3658 The XML attribute `RuleId` in the element `<Rule>`.
- 3659 The XML attribute `MatchId` in the element `<Match>`.
- 3660 The XML attribute `VariableId` in the element `<VariableDefinition>`.
- 3661 The XML attribute `VariableId` in the element `<VariableReference>`.
- 3662 The XML attributes `Category`, `ContextSelectorId` and `DataType` in the element
3663 `<AttributeSelector>`.
- 3664 The XML attributes `Category`, `AttributeId`, `DataType` and `Issuer` in the element
3665 `<AttributeDesignator>`.
- 3666 The XML attribute `DataType` in the element `<AttributeValue>`.
- 3667 The XML attribute `FunctionId` in the element `<Function>`.
- 3668 The XML attribute `FunctionId` in the element `<Apply>`.
- 3669
- 3670 It is RECOMMENDED that extensions to XACML use the same definition of identifier equality for similar
3671 identifiers.
- 3672 It is RECOMMENDED that extensions which define identifiers do not define identifiers which could be
3673 easily misinterpreted by people as being subject to other kind of processing, such as URL character
3674 escaping, before matching.

3675 8 XACML extensibility points (non-normative)

3676 This section describes the points within the XACML model and schema where extensions can be added.

3677 8.1 Extensible XML attribute types

3678 The following XML attributes have values that are URIs. These may be extended by the creation of new
3679 URIs associated with new semantics for these attributes.

3680 *Category*,

3681 *AttributeId*,

3682 *DataType*,

3683 *FunctionId*,

3684 *MatchId*,

3685 *ObligationId*,

3686 *AdviceId*,

3687 *PolicyCombiningAlgId*,

3688 *RuleCombiningAlgId*,

3689 *StatusCode*.

3690 See Section 5 for definitions of these *attribute* types.

3691 8.2 Structured attributes

3692 <AttributeValue> elements MAY contain an instance of a structured XML data-type. Section 7.3.1
3693 describes a number of standard techniques to identify data items within such a structured *attribute*.
3694 Listed here are some additional techniques that require XACML extensions.

- 3695 1. For a given structured data-type, a community of XACML users MAY define new *attribute*
3696 identifiers for each leaf sub-element of the structured data-type that has a type conformant with
3697 one of the XACML-defined primitive data-types. Using these new *attribute* identifiers, the *PEPs*
3698 or *context handlers* used by that community of users can flatten instances of the structured
3699 data-type into a sequence of individual <Attribute> elements. Each such <Attribute>
3700 element can be compared using the XACML-defined functions. Using this method, the structured
3701 data-type itself never appears in an <AttributeValue> element.
- 3702 2. A community of XACML users MAY define a new function that can be used to compare a value of
3703 the structured data-type against some other value. This method may only be used by *PDPs* that
3704 support the new function.

3705 9 Security and privacy considerations (non- 3706 normative)

3707 This section identifies possible security and privacy compromise scenarios that should be considered
3708 when implementing an XACML-based system. The section is informative only. It is left to the
3709 implementer to decide whether these compromise scenarios are practical in their environment and to
3710 select appropriate safeguards.

3711 9.1 Threat model

3712 We assume here that the adversary has access to the communication channel between the XACML
3713 actors and is able to interpret, insert, delete, and modify messages or parts of messages.

3714 Additionally, an actor may use information from a former message maliciously in subsequent transactions.
3715 It is further assumed that *rules* and *policies* are only as reliable as the actors that create and use them.
3716 Thus it is incumbent on each actor to establish appropriate trust in the other actors upon which it relies.
3717 Mechanisms for trust establishment are outside the scope of this specification.

3718 The messages that are transmitted between the actors in the XACML model are susceptible to attack by
3719 malicious third parties. Other points of vulnerability include the *PEP*, the *DDP*, and the *PAP*. While some
3720 of these entities are not strictly within the scope of this specification, their compromise could lead to the
3721 compromise of *access control* enforced by the *PEP*.

3722 It should be noted that there are other components of a distributed system that may be compromised,
3723 such as an operating system and the domain-name system (DNS) that are outside the scope of this
3724 discussion of threat models. Compromise in these components may also lead to a policy violation.

3725 The following sections detail specific compromise scenarios that may be relevant to an XACML system.

3726 9.1.1 Unauthorized disclosure

3727 XACML does not specify any inherent mechanisms to protect the confidentiality of the messages
3728 exchanged between actors. Therefore, an adversary could observe the messages in transit. Under
3729 certain security *policies*, disclosure of this information is a violation. Disclosure of *attributes* or the types
3730 of *decision requests* that a *subject* submits may be a breach of privacy policy. In the commercial
3731 sector, the consequences of unauthorized disclosure of personal data may range from embarrassment to
3732 the custodian, to imprisonment and/or large fines in the case of medical or financial data.

3733 Unauthorized disclosure is addressed by confidentiality safeguards.

3734 9.1.2 Message replay

3735 A message replay attack is one in which the adversary records and replays legitimate messages between
3736 XACML actors. This attack may lead to denial of service, the use of out-of-date information or
3737 impersonation.

3738 Prevention of replay attacks requires the use of message freshness safeguards.

3739 Note that encryption of the message does not mitigate a replay attack since the message is simply
3740 replayed and does not have to be understood by the adversary.

3741 9.1.3 Message insertion

3742 A message insertion attack is one in which the adversary inserts messages in the sequence of messages
3743 between XACML actors.

3744 The solution to a message insertion attack is to use mutual authentication and message sequence
3745 integrity safeguards between the actors. It should be noted that just using SSL mutual authentication is
3746 not sufficient. This only proves that the other party is the one identified by the *subject* of the X.509

3747 certificate. In order to be effective, it is necessary to confirm that the certificate **subject** is authorized to
3748 send the message.

3749 9.1.4 Message deletion

3750 A message deletion attack is one in which the adversary deletes messages in the sequence of messages
3751 between XACML actors. Message deletion may lead to denial of service. However, a properly designed
3752 XACML system should not render an incorrect **authorization decision** as a result of a message deletion
3753 attack.

3754 The solution to a message deletion attack is to use message sequence integrity safeguards between the
3755 actors.

3756 9.1.5 Message modification

3757 If an adversary can intercept a message and change its contents, then they may be able to alter an
3758 **authorization decision**. A message integrity safeguard can prevent a successful message modification
3759 attack.

3760 9.1.6 NotApplicable results

3761 A result of "NotApplicable" means that the **PDP** could not locate a **policy** whose **target** matched the
3762 information in the **decision request**. In general, it is highly recommended that a "Deny" **effect policy** be
3763 used, so that when a **PDP** would have returned "NotApplicable", a result of "Deny" is returned instead.

3764 In some security models, however, such as those found in many web servers, an **authorization decision**
3765 of "NotApplicable" is treated as equivalent to "Permit". There are particular security considerations that
3766 must be taken into account for this to be safe. These are explained in the following paragraphs.

3767 If "NotApplicable" is to be treated as "Permit", it is vital that the matching algorithms used by the **policy** to
3768 match elements in the **decision request** be closely aligned with the data syntax used by the applications
3769 that will be submitting the **decision request**. A failure to match will result in "NotApplicable" and be
3770 treated as "Permit". So an unintended failure to match may allow unintended **access**.

3771 Commercial http responders allow a variety of syntaxes to be treated equivalently. The "%" can be used
3772 to represent characters by hex value. The URL path "/./" provides multiple ways of specifying the same
3773 value. Multiple character sets may be permitted and, in some cases, the same printed character can be
3774 represented by different binary values. Unless the matching algorithm used by the **policy** is sophisticated
3775 enough to catch these variations, unintended **access** may be permitted.

3776 It may be safe to treat "NotApplicable" as "Permit" only in a closed environment where all applications that
3777 formulate a **decision request** can be guaranteed to use the exact syntax expected by the **policies**. In a
3778 more open environment, where **decision requests** may be received from applications that use any legal
3779 syntax, it is strongly recommended that "NotApplicable" NOT be treated as "Permit" unless matching
3780 **rules** have been very carefully designed to match all possible applicable inputs, regardless of syntax or
3781 type variations. Note, however, that according to Section 7.2, a **PEP** must deny **access** unless it
3782 receives an explicit "Permit" **authorization decision**.

3783 9.1.7 Negative rules

3784 A negative **rule** is one that is based on a **predicate** not being "True". If not used with care, negative
3785 **rules** can lead to policy violations, therefore some authorities recommend that they not be used.
3786 However, negative **rules** can be extremely efficient in certain cases, so XACML has chosen to include
3787 them. Nevertheless, it is recommended that they be used with care and avoided if possible.

3788 A common use for negative **rules** is to deny **access** to an individual or subgroup when their membership
3789 in a larger group would otherwise permit them **access**. For example, we might want to write a **rule** that
3790 allows all vice presidents to see the unpublished financial data, except for Joe, who is only a ceremonial
3791 vice president and can be indiscreet in his communications. If we have complete control over the
3792 administration of **subject attributes**, a superior approach would be to define "Vice President" and
3793 "Ceremonial Vice President" as distinct groups and then define **rules** accordingly. However, in some

3794 environments this approach may not be feasible. (It is worth noting in passing that referring to individuals
3795 in **rules** does not scale well. Generally, shared **attributes** are preferred.)

3796 If not used with care, negative **rules** can lead to policy violations in two common cases: when **attributes**
3797 are suppressed and when the base group changes. An example of suppressed **attributes** would be if we
3798 have a **policy** that **access** should be permitted, unless the **subject** is a credit risk. If it is possible that
3799 the **attribute** of being a credit risk may be unknown to the **PDP** for some reason, then unauthorized
3800 **access** may result. In some environments, the **subject** may be able to suppress the publication of
3801 **attributes** by the application of privacy controls, or the server or repository that contains the information
3802 may be unavailable for accidental or intentional reasons.

3803 An example of a changing base group would be if there is a **policy** that everyone in the engineering
3804 department may change software source code, except for secretaries. Suppose now that the department
3805 was to merge with another engineering department and the intent is to maintain the same **policy**.
3806 However, the new department also includes individuals identified as administrative assistants, who ought
3807 to be treated in the same way as secretaries. Unless the **policy** is altered, they will unintentionally be
3808 permitted to change software source code. Problems of this type are easy to avoid when one individual
3809 administers all **policies**, but when administration is distributed, as XACML allows, this type of situation
3810 must be explicitly guarded against.

3811 9.1.8 Denial of service

3812 A denial of service attack is one in which the adversary overloads an XACML actor with excessive
3813 computations or network traffic such that legitimate users cannot access the services provided by the
3814 actor.

3815 The urn:oasis:names:tc:xacml:3.0:function:access-permitted function may lead to hard to predict behavior
3816 in the **PDP**. It is possible that the function is invoked during the recursive invocations of the **PDP** such that
3817 loops are formed. Such loops may in some cases lead to large numbers of requests to be generated
3818 before the **PDP** can detect the loop and abort evaluation. Such loops could cause a denial of service at
3819 the **PDP**, either because of a malicious **policy** or because of a mistake in a **policy**.

3820 9.2 Safeguards

3821 9.2.1 Authentication

3822 Authentication provides the means for one party in a transaction to determine the identity of the other
3823 party in the transaction. Authentication may be in one direction, or it may be bilateral.

3824 Given the sensitive nature of **access control** systems, it is important for a **PEP** to authenticate the
3825 identity of the **PDP** to which it sends **decision requests**. Otherwise, there is a risk that an adversary
3826 could provide false or invalid **authorization decisions**, leading to a policy violation.

3827 It is equally important for a **PDP** to authenticate the identity of the **PEP** and assess the level of trust to
3828 determine what, if any, sensitive data should be passed. One should keep in mind that even simple
3829 "Permit" or "Deny" responses could be exploited if an adversary were allowed to make unlimited requests
3830 to a **PDP**.

3831 Many different techniques may be used to provide authentication, such as co-located code, a private
3832 network, a VPN, or digital signatures. Authentication may also be performed as part of the
3833 communication protocol used to exchange the **contexts**. In this case, authentication may be performed
3834 either at the message level or at the session level.

3835 9.2.2 Policy administration

3836 If the contents of **policies** are exposed outside of the **access control** system, potential **subjects** may
3837 use this information to determine how to gain unauthorized **access**.

3838 To prevent this threat, the repository used for the storage of **policies** may itself require **access control**.
3839 In addition, the <Status> element should be used to return values of missing **attributes** only when
3840 exposure of the identities of those **attributes** will not compromise security.

3841 9.2.3 Confidentiality

3842 Confidentiality mechanisms ensure that the contents of a message can be read only by the desired
3843 recipients and not by anyone else who encounters the message while it is in transit. There are two areas
3844 in which confidentiality should be considered: one is confidentiality during transmission; the other is
3845 confidentiality within a <Policy> element.

3846 9.2.3.1 Communication confidentiality

3847 In some environments it is deemed good practice to treat all data within an **access control** system as
3848 confidential. In other environments, **policies** may be made freely available for distribution, inspection,
3849 and audit. The idea behind keeping **policy** information secret is to make it more difficult for an adversary
3850 to know what steps might be sufficient to obtain unauthorized **access**. Regardless of the approach
3851 chosen, the security of the **access control** system should not depend on the secrecy of the **policy**.

3852 Any security considerations related to transmitting or exchanging XACML <Policy> elements are
3853 outside the scope of the XACML standard. While it is important to ensure that the integrity and
3854 confidentiality of <Policy> elements is maintained when they are exchanged between two parties, it is
3855 left to the implementers to determine the appropriate mechanisms for their environment.

3856 Communications confidentiality can be provided by a confidentiality mechanism, such as SSL. Using a
3857 point-to-point scheme like SSL may lead to other vulnerabilities when one of the end-points is
3858 compromised.

3859 9.2.3.2 Statement level confidentiality

3860 In some cases, an implementation may want to encrypt only parts of an XACML <Policy> element.

3861 The XML Encryption Syntax and Processing Candidate Recommendation from W3C can be used to
3862 encrypt all or parts of an XML document. This specification is recommended for use with XACML.

3863 It should go without saying that if a repository is used to facilitate the communication of cleartext (i.e.,
3864 unencrypted) **policy** between the **PAP** and **PDP**, then a secure repository should be used to store this
3865 sensitive data.

3866 9.2.4 Policy integrity

3867 The XACML **policy** used by the **PDP** to evaluate the request **context** is the heart of the system.

3868 Therefore, maintaining its integrity is essential. There are two aspects to maintaining the integrity of the
3869 **policy**. One is to ensure that <Policy> elements have not been altered since they were originally
3870 created by the **PAP**. The other is to ensure that <Policy> elements have not been inserted or deleted
3871 from the set of **policies**.

3872 In many cases, both aspects can be achieved by ensuring the integrity of the actors and implementing
3873 session-level mechanisms to secure the communication between actors. The selection of the appropriate
3874 mechanisms is left to the implementers. However, when **policy** is distributed between organizations to
3875 be acted on at a later time, or when the **policy** travels with the protected **resource**, it would be useful to
3876 sign the **policy**. In these cases, the XML Signature Syntax and Processing standard from W3C is
3877 recommended to be used with XACML.

3878 Digital signatures should only be used to ensure the integrity of the statements. Digital signatures should
3879 not be used as a method of selecting or evaluating **policy**. That is, the **PDP** should not request a **policy**
3880 based on who signed it or whether or not it has been signed (as such a basis for selection would, itself,
3881 be a matter of policy). However, the **PDP** must verify that the key used to sign the **policy** is one
3882 controlled by the purported **issuer** of the **policy**. The means to do this are dependent on the specific
3883 signature technology chosen and are outside the scope of this document.

3884 9.2.5 Policy identifiers

3885 Since **policies** can be referenced by their identifiers, it is the responsibility of the **PAP** to ensure that
3886 these are unique. Confusion between identifiers could lead to misidentification of the **applicable policy**.

3887 This specification is silent on whether a **PAP** must generate a new identifier when a **policy** is modified or
3888 may use the same identifier in the modified **policy**. This is a matter of administrative practice. However,
3889 care must be taken in either case. If the identifier is reused, there is a danger that other **policies** or
3890 **policy sets** that reference it may be adversely affected. Conversely, if a new identifier is used, these
3891 other **policies** may continue to use the prior **policy**, unless it is deleted. In either case the results may
3892 not be what the **policy** administrator intends.

3893 If a **PDP** is provided with **policies** from distinct sources which might not be fully trusted, as in the use of
3894 the administration profile [**XACMLAdmin**], there is a concern that someone could intentionally publish a
3895 **policy** with an id which collides with another **policy**. This could cause **policy** references that point to the
3896 wrong **policy**, and may cause other unintended consequences in an implementation which is predicated
3897 upon having unique **policy** identifiers.

3898 If this issue is a concern it is RECOMMENDED that distinct **policy** issuers or sources are assigned
3899 distinct namespaces for **policy** identifiers. One method is to make sure that the **policy** identifier begins
3900 with a string which has been assigned to the particular **policy** issuer or source. The remainder of the
3901 **policy** identifier is an issuer-specific unique part. For instance, Alice from Example Inc. could be assigned
3902 the **policy** identifiers which begin with `http://example.com/xacml/policyId/alice/`. The **PDP** or another
3903 trusted component can then verify that the authenticated source of the **policy** is Alice at Example Inc, or
3904 otherwise reject the **policy**. Anyone else will be unable to publish **policies** with identifiers which collide
3905 with the **policies** of Alice.

3906 9.2.6 Trust model

3907 Discussions of authentication, integrity and confidentiality safeguards necessarily assume an underlying
3908 trust model: how can one actor come to believe that a given key is uniquely associated with a specific,
3909 identified actor so that the key can be used to encrypt data for that actor or verify signatures (or other
3910 integrity structures) from that actor? Many different types of trust models exist, including strict
3911 hierarchies, distributed authorities, the Web, the bridge, and so on.

3912 It is worth considering the relationships between the various actors of the **access control** system in terms
3913 of the interdependencies that do and do not exist.

- 3914 • None of the entities of the authorization system are dependent on the **PEP**. They may collect data
3915 from it, (for example authentication data) but are responsible for verifying it themselves.
- 3916 • The correct operation of the system depends on the ability of the **PEP** to actually enforce **policy**
3917 **decisions**.
- 3918 • The **PEP** depends on the **PDP** to correctly evaluate **policies**. This in turn implies that the **PDP** is
3919 supplied with the correct inputs. Other than that, the **PDP** does not depend on the **PEP**.
- 3920 • The **PDP** depends on the **PAP** to supply appropriate **policies**. The **PAP** is not dependent on other
3921 components.

3922 9.2.7 Privacy

3923 It is important to be aware that any transactions that occur with respect to **access control** may reveal
3924 private information about the actors. For example, if an XACML **policy** states that certain data may only
3925 be read by **subjects** with "Gold Card Member" status, then any transaction in which a **subject** is
3926 permitted **access** to that data leaks information to an adversary about the **subject's** status. Privacy
3927 considerations may therefore lead to encryption and/or to **access control** requirements surrounding the
3928 enforcement of XACML **policy** instances themselves: confidentiality-protected channels for the
3929 request/response protocol messages, protection of **subject attributes** in storage and in transit, and so
3930 on.

3931 Selection and use of privacy mechanisms appropriate to a given environment are outside the scope of
3932 XACML. The **decision** regarding whether, how, and when to deploy such mechanisms is left to the
3933 implementers associated with the environment.

3934 **9.3 Unicode security issues**

3935 There are many security considerations related to use of Unicode. An XACML implementation SHOULD
3936 follow the advice given in the relevant version of **[UTR36]**.

3937 **9.4 Identifier equality**

3938 Section 7.20 defines the identifier equality operation for XACML. This definition of equality does not do
3939 any kind of canonicalization or escaping of characters. The identifiers defined in the XACML specification
3940 have been selected to not include any ambiguity regarding these aspects. It is RECOMMENDED that
3941 identifiers defined by extensions also do not introduce any identifiers which might be mistaken for being
3942 subject to processing, like for instance URL character encoding using “%”.

3943 **10 Conformance**

3944 **10.1 Introduction**

3945 The XACML specification addresses the following aspect of conformance:

3946 The XACML specification defines a number of functions, etc. that have somewhat special applications,
3947 therefore they are not required to be implemented in an implementation that claims to conform with the
3948 OASIS standard.

3949 **10.2 Conformance tables**

3950 This section lists those portions of the specification that **MUST** be included in an implementation of a **PDP**
3951 that claims to conform to XACML v3.0. A set of test cases has been created to assist in this process.
3952 These test cases can be located from the OASIS XACML TC Web page. The site hosting the test cases
3953 contains a full description of the test cases and how to execute them.

3954 Note: "M" means mandatory-to-implement. "O" means optional.

3955 The implementation **MUST** follow sections 5, 6, 7, Appendix A, Appendix B and Appendix C where they
3956 apply to implemented items in the following tables.

3957 **10.2.1 Schema elements**

3958 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

3959 **10.2.2 Identifier Prefixes**

3960 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

3961 **10.2.3 Algorithms**

3962 The implementation MUST include the **rule-** and **policy-combining algorithms** associated with the
3963 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

3964 10.2.4 Status Codes

3965 Implementation support for the <StatusCode> element is optional, but if the element is supported, then
3966 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

3967 10.2.5 Attributes

3968 The implementation MUST support the **attributes** associated with the following identifiers as specified by
3969 XACML. If values for these **attributes** are not present in the **decision request**, then their values MUST
3970 be supplied by the **context handler**. So, unlike most other **attributes**, their semantics are not
3971 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

3972 10.2.6 Identifiers

3973 The implementation MUST use the **attributes** associated with the following identifiers in the way XACML
3974 has defined. This requirement pertains primarily to implementations of a **PAP** or **PEP** that uses XACML,
3975 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	O
urn:oasis:names:tc:xacml:1.0:action:action-namespace	O
urn:oasis:names:tc:xacml:1.0:action:implied-action	O

3976 **10.2.7 Data-types**

3977 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

3978 **10.2.8 Functions**

3979 The implementation MUST properly process those functions associated with the identifiers marked with
3980 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

3981 **10.2.9 Identifiers planned for future deprecation**

3982 These identifiers are associated with previous versions of XACML and newer alternatives exist in XACML
3983 3.0. They are planned to be deprecated at some unspecified point in the future. It is RECOMMENDED
3984 that these identifiers not be used in new policies and requests.

3985 The implementation MUST properly process those features associated with the identifiers marked with an
3986 "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

3987

3988 Appendix A. Data-types and functions (normative)

3989 A.1 Introduction

3990 This section specifies the data-types and functions used in XACML to create *predicates* for *conditions*
3991 and *target* matches.

3992 This specification combines the various standards set forth by IEEE and ANSI for string representation of
3993 numeric values, as well as the evaluation of arithmetic functions. It describes the primitive data-types and
3994 *bags*. The standard functions are named and their operational semantics are described.

3995 A.2 Data-types

3996 Although XML instances represent all data-types as strings, an XACML *PDP* must operate on types of
3997 data that, while they have string representations, are not just strings. Types such as Boolean, integer,
3998 and double MUST be converted from their XML string representations to values that can be compared
3999 with values in their domain of discourse, such as numbers. The following primitive data-types are
4000 specified for use with XACML and have explicit data representations:

- 4001 • <http://www.w3.org/2001/XMLSchema#string>
- 4002 • <http://www.w3.org/2001/XMLSchema#boolean>
- 4003 • <http://www.w3.org/2001/XMLSchema#integer>
- 4004 • <http://www.w3.org/2001/XMLSchema#double>
- 4005 • <http://www.w3.org/2001/XMLSchema#time>
- 4006 • <http://www.w3.org/2001/XMLSchema#date>
- 4007 • <http://www.w3.org/2001/XMLSchema#dateTime>
- 4008 • <http://www.w3.org/2001/XMLSchema#anyURI>
- 4009 • <http://www.w3.org/2001/XMLSchema#hexBinary>
- 4010 • <http://www.w3.org/2001/XMLSchema#base64Binary>
- 4011 • <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- 4012 • <http://www.w3.org/2001/XMLSchema#yearMonthDuration>
- 4013 • <urn:oasis:names:tc:xacml:1.0:data-type:x500Name>
- 4014 • <urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name>
- 4015 • <urn:oasis:names:tc:xacml:2.0:data-type:ipAddress>
- 4016 • <urn:oasis:names:tc:xacml:2.0:data-type:dnsName>
- 4017 • <urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression>

4018 For the sake of improved interoperability, it is RECOMMENDED that all time references be in UTC time.

4019 An XACML *PDP* SHALL be capable of converting string representations into various primitive data-types.
4020 For doubles, XACML SHALL use the conversions described in [IEEE754].

4021 XACML defines four data-types representing identifiers for *subjects* or *resources*; these are:

- 4022 “urn:oasis:names:tc:xacml:1.0:data-type:x500Name”,
- 4023 “urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name”
- 4024 “urn:oasis:names:tc:xacml:2.0:data-type:ipAddress” and
- 4025 “urn:oasis:names:tc:xacml:2.0:data-type:dnsName”

4026 These types appear in several standard applications, such as TLS/SSL and electronic mail.

4027 X.500 directory name

4028 The "urn:oasis:names:tc:xacml:1.0:data-type:x500Name" primitive type represents an ITU-T Rec.
4029 X.520 Distinguished Name. The valid syntax for such a name is described in IETF RFC 2253
4030 "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished
4031 Names".

4032 **RFC 822 name**

4033 The "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" primitive type represents an electronic
4034 mail address. The valid syntax for such a name is described in IETF RFC 2821, Section 4.1.2,
4035 Command Argument Syntax, under the term "Mailbox".

4036 **IP address**

4037 The "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" primitive type represents an IPv4 or IPv6
4038 network address, with optional mask and optional port or port range. The syntax SHALL be:

4039 `ipAddress = address ["/" mask] [":" [portrange]]`

4040 For an IPv4 address, the address and mask are formatted in accordance with the syntax for a
4041 "host" in IETF RFC 2396 "Uniform Resource Identifiers (URI): Generic Syntax", section 3.2.

4042 For an IPv6 address, the address and mask are formatted in accordance with the syntax for an
4043 "ipv6reference" in IETF RFC 2732 "Format for Literal IPv6 Addresses in URL's". (Note that an
4044 IPv6 address or mask, in this syntax, is enclosed in literal "[" "]" brackets.)

4045 **DNS name**

4046 The "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" primitive type represents a Domain
4047 Name Service (DNS) host name, with optional port or port range. The syntax SHALL be:

4048 `dnsName = hostname [":" portrange]`

4049 The hostname is formatted in accordance with IETF RFC 2396 "Uniform Resource Identifiers
4050 (URI): Generic Syntax", section 3.2, except that a wildcard "*" may be used in the left-most
4051 component of the hostname to indicate "any subdomain" under the domain specified to its right.

4052 For both the "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" and
4053 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" data-types, the port or port range syntax
4054 SHALL be

4055 `portrange = portnumber | "-"portnumber | portnumber "-"portnumber`

4056 where "portnumber" is a decimal port number. If the port number is of the form "-x", where "x" is
4057 a port number, then the range is all ports numbered "x" and below. If the port number is of the
4058 form "x-", then the range is all ports numbered "x" and above. [This syntax is taken from the Java
4059 SocketPermission.]

4060 **XPath expression**

4061 The "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" primitive type represents an
4062 XPath expression over the XML in a <Content> element. The syntax is defined by the XPath
4063 W3C recommendation. The content of this data type also includes the context in which
4064 namespaces prefixes in the expression are resolved, which distinguishes it from a plain string and
4065 the XACML **attribute** category of the <Content> element to which it applies. When the value is
4066 encoded in an <AttributeValue> element, the namespace context is given by the [in-scope
4067 namespaces] (see [INFOSET]) of the <AttributeValue> element, and an XML attribute called
4068 XPathCategory gives the category of the <Content> element where the expression applies.

4069 The XPath expression MUST be evaluated in a context which is equivalent of a stand alone XML
4070 document with the only child of the <Content> element as the document element. Namespace
4071 declarations which are not "visibly utilized", as defined by [exc-c14n], MAY not be present and
4072 MUST NOT be utilized by the XPath expression. The context node of the XPath expression is the
4073 document node of this stand alone document.

4074 A.3 Functions

4075 XACML specifies the following functions. Unless otherwise specified, if an argument of one of these
4076 functions were to evaluate to "Indeterminate", then the function SHALL be set to "Indeterminate".

4077 Note that in each case an implementation is conformant as long as it produces the same result as is
4078 specified here, regardless of how and in what order the implementation behaves internally.

4079 A.3.1 Equality predicates

4080 The following functions are the equality functions for the various primitive types. Each function for a
4081 particular data-type follows a specified standard convention for that data-type.

- 4082 • urn:oasis:names:tc:xacml:1.0:function:string-equal

4083 This function SHALL take two arguments of data-type
4084 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
4085 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if
4086 the value of both of its arguments are of equal length and each string is determined to be equal.
4087 Otherwise, it SHALL return "False". The comparison SHALL use Unicode codepoint collation, as
4088 defined for the identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.

- 4089 • urn:oasis:names:tc:xacml:3.0:function:string-equal-ignore-case

4090 This function SHALL take two arguments of data-type
4091 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
4092 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be "True" if and only if the
4093 two strings are equal as defined by urn:oasis:names:tc:xacml:1.0:function:string-equal after they
4094 have both been converted to lower case with urn:oasis:names:tc:xacml:1.0:function:string-
4095 normalize-to-lower-case.

- 4096 • urn:oasis:names:tc:xacml:1.0:function:boolean-equal

4097 This function SHALL take two arguments of data-type
4098 "http://www.w3.org/2001/XMLSchema#boolean" and SHALL return an
4099 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if
4100 the arguments are equal. Otherwise, it SHALL return "False".

- 4101 • urn:oasis:names:tc:xacml:1.0:function:integer-equal

4102 This function SHALL take two arguments of data-type
4103 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return an
4104 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if
4105 the two arguments represent the same number.

- 4106 • urn:oasis:names:tc:xacml:1.0:function:double-equal

4107 This function SHALL take two arguments of data-type
4108 "http://www.w3.org/2001/XMLSchema#double" and SHALL return an
4109 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation on doubles
4110 according to IEEE 754 **[IEEE754]**.

- 4111 • urn:oasis:names:tc:xacml:1.0:function:date-equal

4112 This function SHALL take two arguments of data-type
4113 "http://www.w3.org/2001/XMLSchema#date" and SHALL return an
4114 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to
4115 the "op:date-equal" function **[XF]** Section 10.4.9.

- 4116 • urn:oasis:names:tc:xacml:1.0:function:time-equal

4117 This function SHALL take two arguments of data-type
4118 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
4119 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to
4120 the "op:time-equal" function **[XF]** Section 10.4.12.

- 4121 • urn:oasis:names:tc:xacml:1.0:function:dateTime-equal
- 4122 This function SHALL take two arguments of data-type
4123 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an
4124 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to
4125 the "op:dateTime-equal" function **[XF]** Section 10.4.6.
- 4126 • urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-equal
- 4127 This function SHALL take two arguments of data-type
4128 "http://www.w3.org/2001/XMLSchema#dayTimeDuration" and SHALL return an
4129 "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation
4130 according to the "op:duration-equal" function **[XF]** Section 10.4.5. Note that the lexical
4131 representation of each argument MUST be converted to a value expressed in fractional seconds
4132 **[XF]** Section 10.3.2.
- 4133 • urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-equal
- 4134 This function SHALL take two arguments of data-type
4135 "http://www.w3.org/2001/XMLSchema#yearMonthDuration" and SHALL return an
4136 "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation
4137 according to the "op:duration-equal" function **[XF]** Section 10.4.5. Note that the lexical
4138 representation of each argument MUST be converted to a value expressed in fractional seconds
4139 **[XF]** Section 10.3.2.
- 4140 • urn:oasis:names:tc:xacml:1.0:function:anyURI-equal
- 4141 This function SHALL take two arguments of data-type
4142 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return an
4143 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL convert the arguments to
4144 strings with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI and return "True" if and
4145 only if the values of the two arguments are equal on a codepoint-by-codepoint basis.
- 4146 • urn:oasis:names:tc:xacml:1.0:function:x500Name-equal
- 4147 This function SHALL take two arguments of "urn:oasis:names:tc:xacml:1.0:data-type:x500Name"
4148 and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if
4149 and only if each Relative Distinguished Name (RDN) in the two arguments matches. Otherwise,
4150 it SHALL return "False". Two RDNs shall be said to match if and only if the result of the following
4151 operations is "True" .
- 4152 1. Normalize the two arguments according to IETF RFC 2253 "Lightweight Directory Access
4153 Protocol (v3): UTF-8 String Representation of Distinguished Names".
 - 4154 2. If any RDN contains multiple attributeTypeAndValue pairs, re-order the Attribute
4155 ValuePairs in that RDN in ascending order when compared as octet strings (described in
4156 ITU-T Rec. X.690 (1997 E) Section 11.6 "Set-of components").
 - 4157 3. Compare RDNs using the rules in IETF RFC 3280 "Internet X.509 Public Key
4158 Infrastructure Certificate and Certificate Revocation List (CRL) Profile", Section 4.1.2.4
4159 "Issuer".
- 4160 • urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal
- 4161 This function SHALL take two arguments of data-type "urn:oasis:names:tc:xacml:1.0:data-
4162 type:rfc822Name" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It
4163 SHALL return "True" if and only if the two arguments are equal. Otherwise, it SHALL return
4164 "False". An RFC822 name consists of a local-part followed by "@" followed by a domain-part.
4165 The local-part is case-sensitive, while the domain-part (which is usually a DNS host name) is not
4166 case-sensitive. Perform the following operations:
- 4167 1. Normalize the domain-part of each argument to lower case
 - 4168 2. Compare the expressions by applying the function
4169 "urn:oasis:names:tc:xacml:1.0:function:string-equal" to the normalized arguments.

4170 • urn:oasis:names:tc:xacml:1.0:function:hexBinary-equal
4171 This function SHALL take two arguments of data-type
4172 "http://www.w3.org/2001/XMLSchema#hexBinary" and SHALL return an
4173 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet sequences
4174 represented by the value of both arguments have equal length and are equal in a conjunctive,
4175 point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function.
4176 Otherwise, it SHALL return "False". The conversion from the string representation to an octet
4177 sequence SHALL be as specified in [XS] Section 3.2.15.

4178 • urn:oasis:names:tc:xacml:1.0:function:base64Binary-equal
4179 This function SHALL take two arguments of data-type
4180 "http://www.w3.org/2001/XMLSchema#base64Binary" and SHALL return an
4181 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet sequences
4182 represented by the value of both arguments have equal length and are equal in a conjunctive,
4183 point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function.
4184 Otherwise, it SHALL return "False". The conversion from the string representation to an octet
4185 sequence SHALL be as specified in [XS] Section 3.2.16.

4186 A.3.2 Arithmetic functions

4187 All of the following functions SHALL take two arguments of the specified data-type, integer, or double,
4188 and SHALL return an element of integer or double data-type, respectively. However, the "add" and
4189 "multiply" functions MAY take more than two arguments. Each function evaluation operating on doubles
4190 SHALL proceed as specified by their logical counterparts in IEEE 754 [IEEE754]. For all of these
4191 functions, if any argument is "Indeterminate", then the function SHALL evaluate to "Indeterminate". In the
4192 case of the divide functions, if the divisor is zero, then the function SHALL evaluate to "Indeterminate".

4193 • urn:oasis:names:tc:xacml:1.0:function:integer-add
4194 This function MUST accept two or more arguments.

4195 • urn:oasis:names:tc:xacml:1.0:function:double-add
4196 This function MUST accept two or more arguments.

4197 • urn:oasis:names:tc:xacml:1.0:function:integer-subtract
4198 The result is the second argument subtracted from the first argument.

4199 • urn:oasis:names:tc:xacml:1.0:function:double-subtract
4200 The result is the second argument subtracted from the first argument.

4201 • urn:oasis:names:tc:xacml:1.0:function:integer-multiply
4202 This function MUST accept two or more arguments.

4203 • urn:oasis:names:tc:xacml:1.0:function:double-multiply
4204 This function MUST accept two or more arguments.

4205 • urn:oasis:names:tc:xacml:1.0:function:integer-divide
4206 The result is the first argument divided by the second argument.

4207 • urn:oasis:names:tc:xacml:1.0:function:double-divide
4208 The result is the first argument divided by the second argument.

4209 • urn:oasis:names:tc:xacml:1.0:function:integer-mod
4210 The result is remainder of the first argument divided by the second argument.

4211 The following functions SHALL take a single argument of the specified data-type. The round and floor
4212 functions SHALL take a single argument of data-type "http://www.w3.org/2001/XMLSchema#double" and
4213 return a value of the data-type "http://www.w3.org/2001/XMLSchema#double".

4214 • urn:oasis:names:tc:xacml:1.0:function:integer-abs

- 4215 • urn:oasis:names:tc:xacml:1.0:function:double-abs
- 4216 • urn:oasis:names:tc:xacml:1.0:function:round
- 4217 • urn:oasis:names:tc:xacml:1.0:function:floor

4218 **A.3.3 String conversion functions**

4219 The following functions convert between values of the data-type
4220 “http://www.w3.org/2001/XMLSchema#string” primitive types.

- 4221 • urn:oasis:names:tc:xacml:1.0:function:string-normalize-space
 - 4222 This function SHALL take one argument of data-type
 - 4223 “http://www.w3.org/2001/XMLSchema#string” and SHALL normalize the value by stripping off all
 - 4224 leading and trailing white space characters. The whitespace characters are defined in the
 - 4225 metasympol S (Production 3) of [XML].
- 4226 • urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case
 - 4227 This function SHALL take one argument of data-type
 - 4228 “http://www.w3.org/2001/XMLSchema#string” and SHALL normalize the value by converting each
 - 4229 upper case character to its lower case equivalent. Case mapping shall be done as specified for
 - 4230 the fn:lower-case function in [XF] with no tailoring for particular languages or environments.

4231 **A.3.4 Numeric data-type conversion functions**

4232 The following functions convert between the data-type “http://www.w3.org/2001/XMLSchema#integer”
4233 and” http://www.w3.org/2001/XMLSchema#double” primitive types.

- 4234 • urn:oasis:names:tc:xacml:1.0:function:double-to-integer
 - 4235 This function SHALL take one argument of data-type
 - 4236 “http://www.w3.org/2001/XMLSchema#double” and SHALL truncate its numeric value to a whole
 - 4237 number and return an element of data-type “http://www.w3.org/2001/XMLSchema#integer”.
- 4238 • urn:oasis:names:tc:xacml:1.0:function:integer-to-double
 - 4239 This function SHALL take one argument of data-type
 - 4240 “http://www.w3.org/2001/XMLSchema#integer” and SHALL promote its value to an element of
 - 4241 data-type “http://www.w3.org/2001/XMLSchema#double” with the same numeric value. If the
 - 4242 integer argument is outside the range which can be represented by a double, the result SHALL
 - 4243 be Indeterminate, with the status code “urn:oasis:names:tc:xacml:1.0:status:processing-error”.

4244 **A.3.5 Logical functions**

4245 This section contains the specification for logical functions that operate on arguments of data-type
4246 “http://www.w3.org/2001/XMLSchema#boolean”.

- 4247 • urn:oasis:names:tc:xacml:1.0:function:or
 - 4248 This function SHALL return "False" if it has no arguments and SHALL return "True" if at least one
 - 4249 of its arguments evaluates to "True". The order of evaluation SHALL be from first argument to
 - 4250 last. The evaluation SHALL stop with a result of "True" if any argument evaluates to "True",
 - 4251 leaving the rest of the arguments unevaluated.
- 4252 • urn:oasis:names:tc:xacml:1.0:function:and
 - 4253 This function SHALL return "True" if it has no arguments and SHALL return "False" if one of its
 - 4254 arguments evaluates to "False". The order of evaluation SHALL be from first argument to last.
 - 4255 The evaluation SHALL stop with a result of "False" if any argument evaluates to "False", leaving
 - 4256 the rest of the arguments unevaluated.
- 4257 • urn:oasis:names:tc:xacml:1.0:function:n-of
 - 4258 The first argument to this function SHALL be of data-type
 - 4259 http://www.w3.org/2001/XMLSchema#integer. The remaining arguments SHALL be of data-type

4260 <http://www.w3.org/2001/XMLSchema#boolean>. The first argument specifies the minimum
4261 number of the remaining arguments that MUST evaluate to "True" for the expression to be
4262 considered "True". If the first argument is 0, the result SHALL be "True". If the number of
4263 arguments after the first one is less than the value of the first argument, then the expression
4264 SHALL result in "Indeterminate". The order of evaluation SHALL be: first evaluate the integer
4265 value, and then evaluate each subsequent argument. The evaluation SHALL stop and return
4266 "True" if the specified number of arguments evaluate to "True". The evaluation of arguments
4267 SHALL stop if it is determined that evaluating the remaining arguments will not satisfy the
4268 requirement.

4269 • <urn:oasis:names:tc:xacml:1.0:function:not>

4270 This function SHALL take one argument of data-type
4271 "<http://www.w3.org/2001/XMLSchema#boolean>". If the argument evaluates to "True", then the
4272 result of the expression SHALL be "False". If the argument evaluates to "False", then the result
4273 of the expression SHALL be "True".

4274 Note: When evaluating and, or, or n-of, it may not be necessary to attempt a full evaluation of each
4275 argument in order to determine whether the evaluation of the argument would result in "Indeterminate".
4276 Analysis of the argument regarding the availability of its *attributes*, or other analysis regarding errors,
4277 such as "divide-by-zero", may render the argument error free. Such arguments occurring in the
4278 expression in a position after the evaluation is stated to stop need not be processed.

4279 **A.3.6 Numeric comparison functions**

4280 These functions form a minimal set for comparing two numbers, yielding a Boolean result. For doubles
4281 they SHALL comply with the rules governed by IEEE 754 [IEEE754].

- 4282 • <urn:oasis:names:tc:xacml:1.0:function:integer-greater-than>
- 4283 • <urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal>
- 4284 • <urn:oasis:names:tc:xacml:1.0:function:integer-less-than>
- 4285 • <urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal>
- 4286 • <urn:oasis:names:tc:xacml:1.0:function:double-greater-than>
- 4287 • <urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal>
- 4288 • <urn:oasis:names:tc:xacml:1.0:function:double-less-than>
- 4289 • <urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal>

4290 **A.3.7 Date and time arithmetic functions**

4291 These functions perform arithmetic operations with date and time.

4292 • <urn:oasis:names:tc:xacml:3.0:function:date-time-add-day-time-duration>

4293 This function SHALL take two arguments, the first SHALL be of data-type
4294 "<http://www.w3.org/2001/XMLSchema#dateTime>" and the second SHALL be of data-type
4295 "<http://www.w3.org/2001/XMLSchema#dayTimeDuration>". It SHALL return a result of
4296 "<http://www.w3.org/2001/XMLSchema#dateTime>". This function SHALL return the value by
4297 adding the second argument to the first argument according to the specification of adding
4298 durations to date and time [XS] Appendix E.

4299 • <urn:oasis:names:tc:xacml:3.0:function:date-time-add-year-month-duration>

4300 This function SHALL take two arguments, the first SHALL be a
4301 "<http://www.w3.org/2001/XMLSchema#dateTime>" and the second SHALL be a
4302 "<http://www.w3.org/2001/XMLSchema#yearMonthDuration>". It SHALL return a result of
4303 "<http://www.w3.org/2001/XMLSchema#dateTime>". This function SHALL return the value by
4304 adding the second argument to the first argument according to the specification of adding
4305 durations to date and time [XS] Appendix E.

- 4306 • urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-dayTimeDuration
 - 4307 This function SHALL take two arguments, the first SHALL be a
 - 4308 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a
 - 4309 "http://www.w3.org/2001/XMLSchema#dayTimeDuration". It SHALL return a result of
 - 4310 "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration,
 - 4311 then this function SHALL return the value by adding the corresponding negative duration, as per
 - 4312 the specification [XS] Appendix E. If the second argument is a negative duration, then the result
 - 4313 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-
 - 4314 dayTimeDuration" had been applied to the corresponding positive duration.
- 4315 • urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-yearMonthDuration
 - 4316 This function SHALL take two arguments, the first SHALL be a
 - 4317 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a
 - 4318 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
 - 4319 "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration,
 - 4320 then this function SHALL return the value by adding the corresponding negative duration, as per
 - 4321 the specification [XS] Appendix E. If the second argument is a negative duration, then the result
 - 4322 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-
 - 4323 yearMonthDuration" had been applied to the corresponding positive duration.
- 4324 • urn:oasis:names:tc:xacml:3.0:function:date-add-yearMonthDuration
 - 4325 This function SHALL take two arguments, the first SHALL be a
 - 4326 "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a
 - 4327 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
 - 4328 "http://www.w3.org/2001/XMLSchema#date". This function SHALL return the value by adding the
 - 4329 second argument to the first argument according to the specification of adding duration to date
 - 4330 [XS] Appendix E.
- 4331 • urn:oasis:names:tc:xacml:3.0:function:date-subtract-yearMonthDuration
 - 4332 This function SHALL take two arguments, the first SHALL be a
 - 4333 "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a
 - 4334 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
 - 4335 "http://www.w3.org/2001/XMLSchema#date". If the second argument is a positive duration, then
 - 4336 this function SHALL return the value by adding the corresponding negative duration, as per the
 - 4337 specification [XS] Appendix E. If the second argument is a negative duration, then the result
 - 4338 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration"
 - 4339 had been applied to the corresponding positive duration.

4340 A.3.8 Non-numeric comparison functions

4341 These functions perform comparison operations on two arguments of non-numerical types.

- 4342 • urn:oasis:names:tc:xacml:1.0:function:string-greater-than
 - 4343 This function SHALL take two arguments of data-type
 - 4344 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
 - 4345 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
 - 4346 argument is lexicographically strictly greater than the second argument. Otherwise, it SHALL
 - 4347 return "False". The comparison SHALL use Unicode codepoint collation, as defined for the
 - 4348 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [XF].
- 4349 • urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal
 - 4350 This function SHALL take two arguments of data-type
 - 4351 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
 - 4352 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
 - 4353 argument is lexicographically greater than or equal to the second argument. Otherwise, it SHALL
 - 4354 return "False". The comparison SHALL use Unicode codepoint collation, as defined for the
 - 4355 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [XF].

- 4356 • urn:oasis:names:tc:xacml:1.0:function:string-less-than
- 4357 This function SHALL take two arguments of data-type
 4358 “http://www.w3.org/2001/XMLSchema#string” and SHALL return an
 4359 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only the first
 4360 argument is lexicographically strictly less than the second argument. Otherwise, it SHALL return
 4361 “False”. The comparison SHALL use Unicode codepoint collation, as defined for the identifier
 4362 http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.
- 4363 • urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal
- 4364 This function SHALL take two arguments of data-type
 4365 “http://www.w3.org/2001/XMLSchema#string” and SHALL return an
 4366 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only the first
 4367 argument is lexicographically less than or equal to the second argument. Otherwise, it SHALL
 4368 return “False”. The comparison SHALL use Unicode codepoint collation, as defined for the
 4369 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.
- 4370 • urn:oasis:names:tc:xacml:1.0:function:time-greater-than
- 4371 This function SHALL take two arguments of data-type
 4372 “http://www.w3.org/2001/XMLSchema#time” and SHALL return an
 4373 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
 4374 argument is greater than the second argument according to the order relation specified for
 4375 “http://www.w3.org/2001/XMLSchema#time” **[XS]** Section 3.2.8. Otherwise, it SHALL return
 4376 “False”. Note: it is illegal to compare a time that includes a time-zone value with one that does
 4377 not. In such cases, the time-in-range function should be used.
- 4378 • urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal
- 4379 This function SHALL take two arguments of data-type
 4380 “http://www.w3.org/2001/XMLSchema#time” and SHALL return an
 4381 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
 4382 argument is greater than or equal to the second argument according to the order relation
 4383 specified for “http://www.w3.org/2001/XMLSchema#time” **[XS]** Section 3.2.8. Otherwise, it
 4384 SHALL return “False”. Note: it is illegal to compare a time that includes a time-zone value with
 4385 one that does not. In such cases, the time-in-range function should be used.
- 4386 • urn:oasis:names:tc:xacml:1.0:function:time-less-than
- 4387 This function SHALL take two arguments of data-type
 4388 “http://www.w3.org/2001/XMLSchema#time” and SHALL return an
 4389 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
 4390 argument is less than the second argument according to the order relation specified for
 4391 “http://www.w3.org/2001/XMLSchema#time” **[XS]** Section 3.2.8. Otherwise, it SHALL return
 4392 “False”. Note: it is illegal to compare a time that includes a time-zone value with one that does
 4393 not. In such cases, the time-in-range function should be used.
- 4394 • urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal
- 4395 This function SHALL take two arguments of data-type
 4396 “http://www.w3.org/2001/XMLSchema#time” and SHALL return an
 4397 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
 4398 argument is less than or equal to the second argument according to the order relation specified
 4399 for “http://www.w3.org/2001/XMLSchema#time” **[XS]** Section 3.2.8. Otherwise, it SHALL return
 4400 “False”. Note: it is illegal to compare a time that includes a time-zone value with one that does
 4401 not. In such cases, the time-in-range function should be used.
- 4402 • urn:oasis:names:tc:xacml:2.0:function:time-in-range
- 4403 This function SHALL take three arguments of data-type
 4404 “http://www.w3.org/2001/XMLSchema#time” and SHALL return an
 4405 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if the first argument falls
 4406 in the range defined inclusively by the second and third arguments. Otherwise, it SHALL return

4407 “False”. Regardless of its value, the third argument SHALL be interpreted as a time that is equal
4408 to, or later than by less than twenty-four hours, the second argument. If no time zone is provided
4409 for the first argument, it SHALL use the default time zone at the **context handler**. If no time zone
4410 is provided for the second or third arguments, then they SHALL use the time zone from the first
4411 argument.

- 4412 • urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than

4413 This function SHALL take two arguments of data-type
4414 “http://www.w3.org/2001/XMLSchema#dateTime” and SHALL return an
4415 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4416 argument is greater than the second argument according to the order relation specified for
4417 “http://www.w3.org/2001/XMLSchema#dateTime” by [XS] part 2, section 3.2.7. Otherwise, it
4418 SHALL return “False”. Note: if a dateTime value does not include a time-zone value, then an
4419 implicit time-zone value SHALL be assigned, as described in [XS].

- 4420 • urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal

4421 This function SHALL take two arguments of data-type
4422 “http://www.w3.org/2001/XMLSchema#dateTime” and SHALL return an
4423 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4424 argument is greater than or equal to the second argument according to the order relation
4425 specified for “http://www.w3.org/2001/XMLSchema#dateTime” by [XS] part 2, section 3.2.7.
4426 Otherwise, it SHALL return “False”. Note: if a dateTime value does not include a time-zone
4427 value, then an implicit time-zone value SHALL be assigned, as described in [XS].

- 4428 • urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than

4429 This function SHALL take two arguments of data-type
4430 “http://www.w3.org/2001/XMLSchema#dateTime” and SHALL return an
4431 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4432 argument is less than the second argument according to the order relation specified for
4433 “http://www.w3.org/2001/XMLSchema#dateTime” by [XS, part 2, section 3.2.7]. Otherwise, it
4434 SHALL return “False”. Note: if a dateTime value does not include a time-zone value, then an
4435 implicit time-zone value SHALL be assigned, as described in [XS].

- 4436 • urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal

4437 This function SHALL take two arguments of data-type “http://www.w3.org/2001/XMLSchema#
4438 dateTime” and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL
4439 return “True” if and only if the first argument is less than or equal to the second argument
4440 according to the order relation specified for “http://www.w3.org/2001/XMLSchema#dateTime” by
4441 [XS] part 2, section 3.2.7. Otherwise, it SHALL return “False”. Note: if a dateTime value does
4442 not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described
4443 in [XS].

- 4444 • urn:oasis:names:tc:xacml:1.0:function:date-greater-than

4445 This function SHALL take two arguments of data-type
4446 “http://www.w3.org/2001/XMLSchema#date” and SHALL return an
4447 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4448 argument is greater than the second argument according to the order relation specified for
4449 “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9. Otherwise, it SHALL
4450 return “False”. Note: if a date value does not include a time-zone value, then an implicit time-
4451 zone value SHALL be assigned, as described in [XS].

- 4452 • urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal

4453 This function SHALL take two arguments of data-type
4454 “http://www.w3.org/2001/XMLSchema#date” and SHALL return an
4455 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4456 argument is greater than or equal to the second argument according to the order relation
4457 specified for “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9.

4458 Otherwise, it SHALL return “False”. Note: if a date value does not include a time-zone value,
4459 then an implicit time-zone value SHALL be assigned, as described in [XS].

- 4460 • urn:oasis:names:tc:xacml:1.0:function:date-less-than

4461 This function SHALL take two arguments of data-type
4462 “http://www.w3.org/2001/XMLSchema#date” and SHALL return an
4463 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4464 argument is less than the second argument according to the order relation specified for
4465 “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9. Otherwise, it SHALL
4466 return “False”. Note: if a date value does not include a time-zone value, then an implicit time-
4467 zone value SHALL be assigned, as described in [XS].

- 4468 • urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal

4469 This function SHALL take two arguments of data-type
4470 “http://www.w3.org/2001/XMLSchema#date” and SHALL return an
4471 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4472 argument is less than or equal to the second argument according to the order relation specified
4473 for “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9. Otherwise, it
4474 SHALL return “False”. Note: if a date value does not include a time-zone value, then an implicit
4475 time-zone value SHALL be assigned, as described in [XS].

4476 **A.3.9 String functions**

4477 The following functions operate on strings and convert to and from other data types.

- 4478 • urn:oasis:names:tc:xacml:2.0:function:string-concatenate

4479 This function SHALL take two or more arguments of data-type
4480 “http://www.w3.org/2001/XMLSchema#string” and SHALL return a
4481 “http://www.w3.org/2001/XMLSchema#string”. The result SHALL be the concatenation, in order,
4482 of the arguments.

- 4483 • urn:oasis:names:tc:xacml:3.0:function:boolean-from-string

4484 This function SHALL take one argument of data-type
4485 “http://www.w3.org/2001/XMLSchema#string”, and SHALL return an
4486 “http://www.w3.org/2001/XMLSchema#boolean”. The result SHALL be the string converted to a
4487 boolean. If the argument is not a valid lexical representation of a boolean, then the result SHALL
4488 be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- 4489 • urn:oasis:names:tc:xacml:3.0:function:string-from-boolean

4490 This function SHALL take one argument of data-type
4491 “http://www.w3.org/2001/XMLSchema#boolean”, and SHALL return an
4492 “http://www.w3.org/2001/XMLSchema#string”. The result SHALL be the boolean converted to a
4493 string in the canonical form specified in [XS].

- 4494 • urn:oasis:names:tc:xacml:3.0:function:integer-from-string

4495 This function SHALL take one argument of data-type
4496 “http://www.w3.org/2001/XMLSchema#string”, and SHALL return an
4497 “http://www.w3.org/2001/XMLSchema#integer”. The result SHALL be the string converted to an
4498 integer. If the argument is not a valid lexical representation of an integer, then the result SHALL
4499 be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- 4500 • urn:oasis:names:tc:xacml:3.0:function:string-from-integer

4501 This function SHALL take one argument of data-type
4502 “http://www.w3.org/2001/XMLSchema#integer”, and SHALL return an
4503 “http://www.w3.org/2001/XMLSchema#string”. The result SHALL be the integer converted to a
4504 string in the canonical form specified in [XS].

- 4505 • urn:oasis:names:tc:xacml:3.0:function:double-from-string

- 4506 This function SHALL take one argument of data-type
 4507 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4508 "http://www.w3.org/2001/XMLSchema#double". The result SHALL be the string converted to a
 4509 double. If the argument is not a valid lexical representation of a double, then the result SHALL be
 4510 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4511 • urn:oasis:names:tc:xacml:3.0:function:string-from-double

4512 This function SHALL take one argument of data-type
 4513 "http://www.w3.org/2001/XMLSchema#double", and SHALL return an
 4514 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the double converted to a
 4515 string in the canonical form specified in **[XS]**.
 - 4516 • urn:oasis:names:tc:xacml:3.0:function:time-from-string

4517 This function SHALL take one argument of data-type
 4518 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4519 "http://www.w3.org/2001/XMLSchema#time". The result SHALL be the string converted to a time.
 4520 If the argument is not a valid lexical representation of a time, then the result SHALL be
 4521 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4522 • urn:oasis:names:tc:xacml:3.0:function:string-from-time

4523 This function SHALL take one argument of data-type
 4524 "http://www.w3.org/2001/XMLSchema#time", and SHALL return an
 4525 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the time converted to a
 4526 string in the canonical form specified in **[XS]**.
 - 4527 • urn:oasis:names:tc:xacml:3.0:function:date-from-string

4528 This function SHALL take one argument of data-type
 4529 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4530 "http://www.w3.org/2001/XMLSchema#date". The result SHALL be the string converted to a
 4531 date. If the argument is not a valid lexical representation of a date, then the result SHALL be
 4532 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4533 • urn:oasis:names:tc:xacml:3.0:function:string-from-date

4534 This function SHALL take one argument of data-type
 4535 "http://www.w3.org/2001/XMLSchema#date", and SHALL return an
 4536 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the date converted to a
 4537 string in the canonical form specified in **[XS]**.
 - 4538 • urn:oasis:names:tc:xacml:3.0:function:dateTime-from-string

4539 This function SHALL take one argument of data-type
 4540 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4541 "http://www.w3.org/2001/XMLSchema#dateTime". The result SHALL be the string converted to a
 4542 dateTime. If the argument is not a valid lexical representation of a dateTime, then the result
 4543 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4544 urn:oasis:names:tc:xacml:3.0:function:string-from-dateTime

4545 This function SHALL take one argument of data-type
 4546 "http://www.w3.org/2001/XMLSchema#dateTime", and SHALL return an
 4547 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the dateTime converted to a
 4548 string in the canonical form specified in **[XS]**.
 - 4549 • urn:oasis:names:tc:xacml:3.0:function:anyURI-from-string

4550 This function SHALL take one argument of data-type
 4551 "http://www.w3.org/2001/XMLSchema#string", and SHALL return a
 4552 "http://www.w3.org/2001/XMLSchema#anyURI". The result SHALL be the URI constructed by
 4553 converting the argument to an URI. If the argument is not a valid lexical representation of a URI,
 4554 then the result SHALL be Indeterminate with status code
 4555 urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- 4556 • urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI
 4557 This function SHALL take one argument of data-type
 4558 "http://www.w3.org/2001/XMLSchema#anyURI", and SHALL return an
 4559 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the URI converted to a
 4560 string in the form it was originally represented in XML form.
- 4561 • urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-from-string
 4562 This function SHALL take one argument of data-type
 4563 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4564 "http://www.w3.org/2001/XMLSchema#dayTimeDuration ". The result SHALL be the string
 4565 converted to a dayTimeDuration. If the argument is not a valid lexical representation of a
 4566 dayTimeDuration, then the result SHALL be Indeterminate with status code
 4567 urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4568 • urn:oasis:names:tc:xacml:3.0:function:string-from-dayTimeDuration
 4569 This function SHALL take one argument of data-type
 4570 "http://www.w3.org/2001/XMLSchema#dayTimeDuration ", and SHALL return an
 4571 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the dayTimeDuration
 4572 converted to a string in the canonical form specified in **[XPathFunc]**.
- 4573 • urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-from-string
 4574 This function SHALL take one argument of data-type
 4575 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4576 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". The result SHALL be the string
 4577 converted to a yearMonthDuration. If the argument is not a valid lexical representation of a
 4578 yearMonthDuration, then the result SHALL be Indeterminate with status code
 4579 urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4580 • urn:oasis:names:tc:xacml:3.0:function:string-from-yearMonthDuration
 4581 This function SHALL take one argument of data-type
 4582 "http://www.w3.org/2001/XMLSchema#yearMonthDuration", and SHALL return an
 4583 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the yearMonthDuration
 4584 converted to a string in the canonical form specified in **[XPathFunc]**.
- 4585 • urn:oasis:names:tc:xacml:3.0:function:x500Name-from-string
 4586 This function SHALL take one argument of data-type
 4587 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4588 "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". The result SHALL be the string converted
 4589 to an x500Name. If the argument is not a valid lexical representation of a X500Name, then the
 4590 result SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4591 • urn:oasis:names:tc:xacml:3.0:function:string-from-x500Name
 4592 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:1.0:data-
 4593 type:x500Name", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result
 4594 SHALL be the x500Name converted to a string in the form it was originally represented in XML
 4595 form..
- 4596 • urn:oasis:names:tc:xacml:3.0:function:rfc822Name-from-string
 4597 This function SHALL take one argument of data-type
 4598 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4599 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". The result SHALL be the string converted
 4600 to an rfc822Name. If the argument is not a valid lexical representation of an rfc822Name, then the
 4601 result SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4602 • urn:oasis:names:tc:xacml:3.0:function:string-from-rfc822Name
 4603 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:1.0:data-
 4604 type:rfc822Name", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The

- 4605 result SHALL be the rfc822Name converted to a string in the form it was originally represented in
4606 XML form.
- 4607 • urn:oasis:names:tc:xacml:3.0:function:ipAddress-from-string

4608 This function SHALL take one argument of data-type
4609 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
4610 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". The result SHALL be the string converted to
4611 an ipAddress. If the argument is not a valid lexical representation of an ipAddress, then the result
4612 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4613 • urn:oasis:names:tc:xacml:3.0:function:string-from-ipAddress

4614 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:2.0:data-
4615 type:ipAddress", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result
4616 SHALL be the ipAddress converted to a string in the form it was originally represented in XML
4617 form.
 - 4618 • urn:oasis:names:tc:xacml:3.0:function:dnsName-from-string

4619 This function SHALL take one argument of data-type
4620 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
4621 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". The result SHALL be the string converted to
4622 a dnsName. If the argument is not a valid lexical representation of a dnsName, then the result
4623 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4624 • urn:oasis:names:tc:xacml:3.0:function:string-from-dnsName

4625 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:2.0:data-
4626 type:dnsName", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result
4627 SHALL be the dnsName converted to a string in the form it was originally represented in XML
4628 form.
 - 4629 • urn:oasis:names:tc:xacml:3.0:function:string-starts-with

4630 This function SHALL take two arguments of data-type
4631 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a
4632 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string
4633 begins with the first string, and false otherwise. Equality testing SHALL be done as defined for
4634 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4635 • urn:oasis:names:tc:xacml:3.0:function:anyURI-starts-with

4636 This function SHALL take a first argument of data-
4637 type"http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type
4638 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a
4639 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted
4640 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI begins with the string,
4641 and false otherwise. Equality testing SHALL be done as defined for
4642 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4643 • urn:oasis:names:tc:xacml:3.0:function:string-ends-with

4644 This function SHALL take two arguments of data-type
4645 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a
4646 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string
4647 ends with the first string, and false otherwise. Equality testing SHALL be done as defined for
4648 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4649 • urn:oasis:names:tc:xacml:3.0:function:anyURI-ends-with

4650 This function SHALL take a first argument of data-type
4651 "http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type
4652 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a
4653 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted
4654 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI ends with the string,

- 4655 and false otherwise. Equality testing SHALL be done as defined for
 4656 urn:oasis:names:tc:xacml:1.0:function:string-equal.
- 4657 • urn:oasis:names:tc:xacml:3.0:function:string-contains
 4658 This function SHALL take two arguments of data-type
 4659 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a
 4660 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string
 4661 contains the first string, and false otherwise. Equality testing SHALL be done as defined for
 4662 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4663 • urn:oasis:names:tc:xacml:3.0:function:anyURI-contains
 4664 This function SHALL take a first argument of data-type
 4665 "http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type
 4666 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a
 4667 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted
 4668 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI contains the string, and
 4669 false otherwise. Equality testing SHALL be done as defined for
 4670 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4671 • urn:oasis:names:tc:xacml:3.0:function:string-substring
 4672 This function SHALL take a first argument of data-type
 4673 "http://www.w3.org/2001/XMLSchema#string" and a second and a third argument of type
 4674 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return a
 4675 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the substring of the first
 4676 argument beginning at the position given by the second argument and ending at the position
 4677 before the position given by the third argument. The first character of the string has position zero.
 4678 The negative integer value -1 given for the third arguments indicates the end of the string. If the
 4679 second or third arguments are out of bounds, then the function MUST evaluate to Indeterminate
 4680 with a status code of urn:oasis:names:tc:xacml:1.0:status:processing-error.
 - 4681 • urn:oasis:names:tc:xacml:3.0:function:anyURI-substring
 4682 This function SHALL take a first argument of data-type
 4683 "http://www.w3.org/2001/XMLSchema#anyURI" and a second and a third argument of type
 4684 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return a
 4685 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the substring of the first
 4686 argument converted to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI
 4687 beginning at the position given by the second argument and ending at the position before the
 4688 position given by the third argument. The first character of the URI converted to a string has
 4689 position zero. The negative integer value -1 given for the third arguments indicates the end of the
 4690 string. If the second or third arguments are out of bounds, then the function MUST evaluate to
 4691 Indeterminate with a status code of
 4692 urn:oasis:names:tc:xacml:1.0:status:processing-error. If the resulting substring
 4693 is not syntactically a valid URI, then the function MUST evaluate to Indeterminate with a status
 4694 code of urn:oasis:names:tc:xacml:1.0:status:processing-error.

4695

4696 A.3.10 Bag functions

4697 These functions operate on a **bag** of 'type' values, where type is one of the primitive data-types, and x.x
 4698 is a version of XACML where the function has been defined. Some additional conditions defined for
 4699 each function below SHALL cause the expression to evaluate to "Indeterminate".

- 4700 • urn:oasis:names:tc:xacml:x.x:function:type-one-and-only
 4701 This function SHALL take a **bag** of 'type' values as an argument and SHALL return a value of
 4702 'type'. It SHALL return the only value in the **bag**. If the **bag** does not have one and only one
 4703 value, then the expression SHALL evaluate to "Indeterminate".

- 4704 • urn:oasis:names:tc:xacml:x.x:function:type-bag-size
- 4705 This function SHALL take a **bag** of 'type' values as an argument and SHALL return an
- 4706 "http://www.w3.org/2001/XMLSchema#integer" indicating the number of values in the **bag**.
- 4707 • urn:oasis:names:tc:xacml:x.x:function:type-is-in
- 4708 This function SHALL take an argument of 'type' as the first argument and a **bag** of 'type' values
- 4709 as the second argument and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean".
- 4710 The function SHALL evaluate to "True" if and only if the first argument matches by the
- 4711 "urn:oasis:names:tc:xacml:x.x:function:type-equal" any value in the **bag**. Otherwise, it SHALL
- 4712 return "False".
- 4713 • urn:oasis:names:tc:xacml:x.x:function:type-bag
- 4714 This function SHALL take any number of arguments of 'type' and return a **bag** of 'type' values
- 4715 containing the values of the arguments. An application of this function to zero arguments SHALL
- 4716 produce an empty **bag** of the specified data-type.

4717 **A.3.11 Set functions**

4718 These functions operate on **bags** mimicking sets by eliminating duplicate elements from a **bag**.

- 4719 • urn:oasis:names:tc:xacml:x.x:function:type-intersection
- 4720 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4721 **bag** of 'type' values such that it contains only elements that are common between the two **bags**,
- 4722 which is determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal". No duplicates, as
- 4723 determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal", SHALL exist in the result.
- 4724 • urn:oasis:names:tc:xacml:x.x:function:type-at-least-one-member-of
- 4725 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4726 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL evaluate to "True" if and
- 4727 only if at least one element of the first argument is contained in the second argument as
- 4728 determined by "urn:oasis:names:tc:xacml:x.x:function:type-is-in".
- 4729 • urn:oasis:names:tc:xacml:x.x:function:type-union
- 4730 This function SHALL take two or more arguments that are both a **bag** of 'type' values. The
- 4731 expression SHALL return a **bag** of 'type' such that it contains all elements of all the argument
- 4732 **bags**. No duplicates, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal",
- 4733 SHALL exist in the result.
- 4734 • urn:oasis:names:tc:xacml:x.x:function:type-subset
- 4735 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4736 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
- 4737 argument is a subset of the second argument. Each argument SHALL be considered to have had
- 4738 its duplicates removed, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal",
- 4739 before the subset calculation.
- 4740 • urn:oasis:names:tc:xacml:x.x:function:type-set-equals
- 4741 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4742 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return the result of applying
- 4743 "urn:oasis:names:tc:xacml:1.0:function:and" to the application of
- 4744 "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the first and second arguments and the
- 4745 application of "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the second and first
- 4746 arguments.

4747 **A.3.12 Higher-order bag functions**

4748 This section describes functions in XACML that perform operations on **bags** such that functions may be

4749 applied to the **bags** in general.

4750 • urn:oasis:names:tc:xacml:3.0:function:any-of

4751 This function applies a Boolean function between specific primitive values and a **bag** of values,
4752 and SHALL return "True" if and only if the **predicate** is "True" for at least one element of the **bag**.

4753 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL
4754 be an <Function> element that names a Boolean function that takes n arguments of primitive
4755 types. Under the remaining n arguments, n-1 parameters SHALL be values of primitive data-
4756 types and one SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated as
4757 if the function named in the <Function> argument were applied to the n-1 non-bag arguments
4758 and each element of the bag argument and the results are combined with
4759 "urn:oasis:names:tc:xacml:1.0:function:or".

4760 For example, the following expression SHALL return "True":

```
4761 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">  
4762   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>  
4763   <AttributeValue  
4764     DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4765   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4766     <AttributeValue  
4767       DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>  
4768     <AttributeValue  
4769       DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4770     <AttributeValue  
4771       DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>  
4772     <AttributeValue  
4773       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4774   </Apply>  
4775 </Apply>
```

4776 This expression is "True" because the first argument is equal to at least one of the elements of
4777 the **bag**, according to the function.

4778 • urn:oasis:names:tc:xacml:3.0:function:all-of

4779 This function applies a Boolean function between a specific primitive value and a **bag** of values,
4780 and returns "True" if and only if the **predicate** is "True" for every element of the **bag**.

4781 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL
4782 be a <Function> element that names a Boolean function that takes n arguments of primitive
4783 types. Under the remaining n arguments, n-1 parameters SHALL be values of primitive data-
4784 types and one SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated as
4785 if the function named in the <Function> argument were applied to the n-1 non-bag arguments
4786 and each element of the bag argument and the results are combined with
4787 "urn:oasis:names:tc:xacml:1.0:function:and".

4788 For example, the following expression SHALL evaluate to "True":

```
4789 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:all-of">  
4790   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4791     greater-than"/>  
4792   <AttributeValue  
4793     DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>  
4794   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4795     <AttributeValue  
4796       DataType="http://www.w3.org/2001/XMLSchema#integer">9</AttributeValue>  
4797     <AttributeValue  
4798       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>  
4799     <AttributeValue  
4800       DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>  
4801     <AttributeValue  
4802       DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>  
4803   </Apply>  
4804 </Apply>
```

4805 This expression is "True" because the first argument (10) is greater than all of the elements of the
4806 **bag** (9,3,4 and 2).

4807 • urn:oasis:names:tc:xacml:3.0:function:any-of-any

4808 This function applies a Boolean function on each tuple from the cross product on all bags
4809 arguments, and returns "True" if and only if the **predicate** is "True" for at least one inside-function
4810 call.

4811 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL
4812 be an <Function> element that names a Boolean function that takes n arguments. The
4813 remaining arguments are either primitive data types or bags of primitive types. The expression
4814 SHALL be evaluated as if the function named in the <Function> argument was applied between
4815 every tuple of the cross product on all bags and the primitive values, and the results were
4816 combined using "urn:oasis:names:tc:xacml:1.0:function:or". The semantics are that the result of
4817 the expression SHALL be "True" if and only if the applied **predicate** is "True" for at least one
4818 function call on the tuples from the **bags** and primitive values.

4819 For example, the following expression SHALL evaluate to "True":

```
4820 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of-any">  
4821   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>  
4822   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4823     <AttributeValue  
4824       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4825     <AttributeValue  
4826       DataType="http://www.w3.org/2001/XMLSchema#string">Mary</AttributeValue>  
4827   </Apply>  
4828   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4829     <AttributeValue  
4830       DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>  
4831     <AttributeValue  
4832       DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4833     <AttributeValue  
4834       DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>  
4835     <AttributeValue  
4836       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4837   </Apply>  
4838 </Apply>
```

4839 This expression is "True" because at least one of the elements of the first **bag**, namely "Ringo", is
4840 equal to at least one of the elements of the second **bag**.

4841 • urn:oasis:names:tc:xacml:1.0:function:all-of-any

4842 This function applies a Boolean function between the elements of two **bags**. The expression
4843 SHALL be "True" if and only if the supplied **predicate** is "True" between each element of the first
4844 **bag** and any element of the second **bag**.

4845 This function SHALL take three arguments. The first argument SHALL be an <Function>
4846 element that names a Boolean function that takes two arguments of primitive types. The second
4847 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a
4848 primitive data-type. The expression SHALL be evaluated as if the
4849 "urn:oasis:names:tc:xacml:3.0:function:any-of" function had been applied to each value of the first
4850 **bag** and the whole of the second **bag** using the supplied xacml:Function, and the results were
4851 then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

4852 For example, the following expression SHALL evaluate to "True":

```
4853 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-any">  
4854   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4855   greater-than"/>  
4856   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4857     <AttributeValue  
4858       DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
```

```

4859         <AttributeValue
4860         DataType="http://www.w3.org/2001/XMLSchema#integer">20</AttributeValue>
4861     </Apply>
4862     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4863         <AttributeValue
4864         DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4865         <AttributeValue
4866         DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4867         <AttributeValue
4868         DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4869         <AttributeValue
4870         DataType="http://www.w3.org/2001/XMLSchema#integer">19</AttributeValue>
4871     </Apply>
4872 </Apply>

```

4873 This expression is "True" because each of the elements of the first **bag** is greater than at least
4874 one of the elements of the second **bag**.

4875 • urn:oasis:names:tc:xacml:1.0:function:any-of-all

4876 This function applies a Boolean function between the elements of two **bags**. The expression
4877 SHALL be "True" if and only if the supplied **predicate** is "True" between each element of the
4878 second **bag** and any element of the first **bag**.

4879 This function SHALL take three arguments. The first argument SHALL be an <Function>
4880 element that names a Boolean function that takes two arguments of primitive types. The second
4881 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a
4882 primitive data-type. The expression SHALL be evaluated as if the
4883 "urn:oasis:names:tc:xacml:3.0:function:any-of" function had been applied to each value of the
4884 second **bag** and the whole of the first **bag** using the supplied xacml:Function, and the results
4885 were then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

4886 For example, the following expression SHALL evaluate to "True":

```

4887 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-all">
4888     <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
4889     greater-than"/>
4890     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4891         <AttributeValue
4892         DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4893         <AttributeValue
4894         DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4895     </Apply>
4896     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4897         <AttributeValue
4898         DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4899         <AttributeValue
4900         DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
4901         <AttributeValue
4902         DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4903         <AttributeValue
4904         DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
4905     </Apply>
4906 </Apply>

```

4907 This expression is "True" because, for all of the values in the second **bag**, there is a value in the
4908 first **bag** that is greater.

4909 • urn:oasis:names:tc:xacml:1.0:function:all-of-all

4910 This function applies a Boolean function between the elements of two **bags**. The expression
4911 SHALL be "True" if and only if the supplied **predicate** is "True" between each and every element
4912 of the first **bag** collectively against all the elements of the second **bag**.

4913 This function SHALL take three arguments. The first argument SHALL be an <Function>
4914 element that names a Boolean function that takes two arguments of primitive types. The second

4915 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a
4916 primitive data-type. The expression is evaluated as if the function named in the <Function>
4917 element were applied between every element of the second argument and every element of the
4918 third argument and the results were combined using "urn:oasis:names:tc:xacml:1.0:function:and".
4919 The semantics are that the result of the expression is "True" if and only if the applied **predicate** is
4920 "True" for all elements of the first **bag** compared to all the elements of the second **bag**.

4921 For example, the following expression SHALL evaluate to "True":

```
4922 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-all">  
4923   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4924   greater-than"/>  
4925   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4926     <AttributeValue  
4927     DataType="http://www.w3.org/2001/XMLSchema#integer">6</AttributeValue>  
4928     <AttributeValue  
4929     DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>  
4930   </Apply>  
4931   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4932     <AttributeValue  
4933     DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>  
4934     <AttributeValue  
4935     DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>  
4936     <AttributeValue  
4937     DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>  
4938     <AttributeValue  
4939     DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>  
4940   </Apply>  
4941 </Apply>
```

4942 This expression is "True" because all elements of the first **bag**, "5" and "6", are each greater than
4943 all of the integer values "1", "2", "3", "4" of the second **bag**.

4944 • urn:oasis:names:tc:xacml:3.0:function:map

4945 This function converts a **bag** of values to another **bag** of values.

4946 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL
4947 be a <Function> element naming a function that takes a n arguments of a primitive data-type
4948 and returns a value of a primitive data-type Under the remaining n arguments, n-1 parameters
4949 SHALL be values of primitive data-types and one SHALL be a **bag** of a primitive data-type. The
4950 expression SHALL be evaluated as if the function named in the <Function> argument were
4951 applied to the n-1 non-bag arguments and each element of the bag argument and resulting in a
4952 **bag** of the converted value. The result SHALL be a **bag** of the primitive data-type that is returned
4953 by the function named in the <xacml:Function> element.

4954 For example, the following expression,

```
4955 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:map">  
4956   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-  
4957   normalize-to-lower-case">  
4958   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4959     <AttributeValue  
4960     DataType="http://www.w3.org/2001/XMLSchema#string">Hello</AttributeValue>  
4961     <AttributeValue  
4962     DataType="http://www.w3.org/2001/XMLSchema#string">World!</AttributeValue>  
4963   </Apply>  
4964 </Apply>
```

4965 evaluates to a **bag** containing "hello" and "world!".

4966 A.3.13 Regular-expression-based functions

4967 These functions operate on various types using regular expressions and evaluate to
4968 "http://www.w3.org/2001/XMLSchema#boolean".

- 4969 • urn:oasis:names:tc:xacml:1.0:function:string-regexp-match
- 4970 This function decides a regular expression match. It SHALL take two arguments of
4971 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
4972 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
4973 expression and the second argument SHALL be a general string. The function specification
4974 SHALL be that of the "xf:matches" function with the arguments reversed [XF] Section 7.6.2.
- 4975 • urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match
- 4976 This function decides a regular expression match. It SHALL take two arguments; the first is of
4977 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
4978 "http://www.w3.org/2001/XMLSchema#anyURI". It SHALL return an
4979 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
4980 expression and the second argument SHALL be a URI. The function SHALL convert the second
4981 argument to type "http://www.w3.org/2001/XMLSchema#string" with
4982 urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI, then apply
4983 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 4984 • urn:oasis:names:tc:xacml:2.0:function:ipAddress-regexp-match
- 4985 This function decides a regular expression match. It SHALL take two arguments; the first is of
4986 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
4987 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". It SHALL return an
4988 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
4989 expression and the second argument SHALL be an IPv4 or IPv6 address. The function SHALL
4990 convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with
4991 urn:oasis:names:tc:xacml:3.0:function:string-from-ipAddress, then apply
4992 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 4993 • urn:oasis:names:tc:xacml:2.0:function:dnsName-regexp-match
- 4994 This function decides a regular expression match. It SHALL take two arguments; the first is of
4995 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
4996 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". It SHALL return an
4997 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
4998 expression and the second argument SHALL be a DNS name. The function SHALL convert the
4999 second argument to type "http://www.w3.org/2001/XMLSchema#string" with
5000 urn:oasis:names:tc:xacml:3.0:function:string-from-dnsName, then apply
5001 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5002 • urn:oasis:names:tc:xacml:2.0:function:rfc822Name-regexp-match
- 5003 This function decides a regular expression match. It SHALL take two arguments; the first is of
5004 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
5005 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". It SHALL return an
5006 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
5007 expression and the second argument SHALL be an RFC 822 name. The function SHALL convert
5008 the second argument to type "http://www.w3.org/2001/XMLSchema#string" with
5009 urn:oasis:names:tc:xacml:3.0:function:string-from-rfc822Name, then apply
5010 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5011 • urn:oasis:names:tc:xacml:2.0:function:x500Name-regexp-match
- 5012 This function decides a regular expression match. It SHALL take two arguments; the first is of
5013 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
5014 "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". It SHALL return an
5015 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
5016 expression and the second argument SHALL be an X.500 directory name. The function SHALL
5017 convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with
5018 urn:oasis:names:tc:xacml:3.0:function:string-from-x500Name, then apply
5019 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

5020 **A.3.14 Special match functions**

5021 These functions operate on various types and evaluate to
5022 "http://www.w3.org/2001/XMLSchema#boolean" based on the specified standard matching algorithm.

- 5023 • urn:oasis:names:tc:xacml:1.0:function:x500Name-match

5024 This function shall take two arguments of "urn:oasis:names:tc:xacml:1.0:data-type:x500Name"
5025 and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and
5026 only if the first argument matches some terminal sequence of RDNs from the second argument
5027 when compared using x500Name-equal.

5028 As an example (non-normative), if the first argument is "O=Medico Corp,C=US" and the second
5029 argument is "cn=John Smith,o=Medico Corp, c=US", then the function will return "True".

- 5030 • urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match

5031 This function SHALL take two arguments, the first is of data-type
5032 "http://www.w3.org/2001/XMLSchema#string" and the second is of data-type
5033 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" and SHALL return an
5034 "http://www.w3.org/2001/XMLSchema#boolean". This function SHALL evaluate to "True" if the
5035 first argument matches the second argument according to the following specification.

5036 An RFC822 name consists of a local-part followed by "@" followed by a domain-part. The local-
5037 part is case-sensitive, while the domain-part (which is usually a DNS name) is not case-sensitive.

5038 The second argument contains a complete rfc822Name. The first argument is a complete or
5039 partial rfc822Name used to select appropriate values in the second argument as follows.

5040 In order to match a particular address in the second argument, the first argument must specify the
5041 complete mail address to be matched. For example, if the first argument is
5042 "Anderson@sun.com", this matches a value in the second argument of "Anderson@sun.com"
5043 and "Anderson@SUN.COM", but not "Anne.Anderson@sun.com", "anderson@sun.com" or
5044 "Anderson@east.sun.com".

5045 In order to match any address at a particular domain in the second argument, the first argument
5046 must specify only a domain name (usually a DNS name). For example, if the first argument is
5047 "sun.com", this matches a value in the second argument of "Anderson@sun.com" or
5048 "Baxter@SUN.COM", but not "Anderson@east.sun.com".

5049 In order to match any address in a particular domain in the second argument, the first argument
5050 must specify the desired domain-part with a leading ".". For example, if the first argument is
5051 ".east.sun.com", this matches a value in the second argument of "Anderson@east.sun.com" and
5052 "anne.anderson@ISRG.EAST.SUN.COM" but not "Anderson@sun.com".

5053 **A.3.15 XPath-based functions**

5054 This section specifies functions that take XPath expressions for arguments. An XPath expression
5055 evaluates to a node-set, which is a set of XML nodes that match the expression. A node or node-set is
5056 not in the formal data-type system of XACML. All comparison or other operations on node-sets are
5057 performed in isolation of the particular function specified. The context nodes and namespace mappings
5058 of the XPath expressions are defined by the XPath data-type, see section B.3. The following functions
5059 are defined:

- 5060 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-count

5061 This function SHALL take an "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" as an
5062 argument and evaluates to an "http://www.w3.org/2001/XMLSchema#integer". The value
5063 returned from the function SHALL be the count of the nodes within the node-set that match the
5064 given XPath expression. If the <Content> element of the category to which the XPath
5065 expression applies to is not present in the request, this function SHALL return a value of zero.

- 5066 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-equal

5067 This function SHALL take two “urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression”
5068 arguments and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. The function
5069 SHALL return "True" if any of the XML nodes in the node-set matched by the first argument
5070 equals any of the XML nodes in the node-set matched by the second argument. Two nodes are
5071 considered equal if they have the same identity. If the <Content> element of the category to
5072 which either XPath expression applies to is not present in the request, this function SHALL return
5073 a value of “False”.

5074 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-match

5075 This function SHALL take two “urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression”
5076 arguments and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. This function
5077 SHALL evaluate to "True" if one of the following two conditions is satisfied: (1) Any of the XML
5078 nodes in the node-set matched by the first argument is equal to any of the XML nodes in the
5079 node-set matched by the second argument; (2) any node below any of the XML nodes in the
5080 node-set matched by the first argument is equal to any of the XML nodes in the node-set
5081 matched by the second argument. Two nodes are considered equal if they have the same
5082 identity. If the <Content> element of the category to which either XPath expression applies to is
5083 not present in the request, this function SHALL return a value of “False”.

5084 NOTE: The first **condition** is equivalent to "xpath-node-equal", and guarantees that "xpath-node-equal" is
5085 a special case of "xpath-node-match".

5086 A.3.16 Other functions

5087 • urn:oasis:names:tc:xacml:3.0:function:access-permitted

5088 This function SHALL take an “http://www.w3.org/2001/XMLSchema#anyURI” and an
5089 “http://www.w3.org/2001/XMLSchema#string” as arguments. The first argument SHALL be
5090 interpreted as an **attribute** category. The second argument SHALL be interpreted as the XML
5091 content of an <Attributes> element with *Category* equal to the first argument. The function
5092 evaluates to an “http://www.w3.org/2001/XMLSchema#boolean”. This function SHALL return
5093 "True" if and only if the **policy** evaluation described below returns the value of "Permit".

5094 The following evaluation is described as if the **context** is actually instantiated, but it is only
5095 required that an equivalent result be obtained.

5096 The function SHALL construct a new **context**, by copying all the information from the current
5097 **context**, omitting any <Attributes> element with *Category* equal to the first argument. The
5098 second function argument SHALL be added to the **context** as the content of an <Attributes>
5099 element with *Category* equal to the first argument.

5100 The function SHALL invoke a complete **policy** evaluation using the newly constructed **context**.
5101 This evaluation SHALL be completely isolated from the evaluation which invoked the function, but
5102 shall use all current **policies** and combining algorithms, including any per request **policies**.

5103 The **PDP** SHALL detect any loop which may occur if successive evaluations invoke this function
5104 by counting the number of total invocations of any instance of this function during any single initial
5105 invocation of the **PDP**. If the total number of invocations exceeds the bound for such invocations,
5106 the initial invocation of this function evaluates to Indeterminate with a
5107 “urn:oasis:names:tc:xacml:1.0:status:processing-error” status code. Also, see the security
5108 considerations in section 9.1.8.

5109 A.3.17 Extension functions and primitive types

5110 Functions and primitive types are specified by string identifiers allowing for the introduction of functions in
5111 addition to those specified by XACML. This approach allows one to extend the XACML module with
5112 special functions and special primitive data-types.

5113 In order to preserve the integrity of the XACML evaluation strategy, the result of an extension function
5114 SHALL depend only on the values of its arguments. Global and hidden parameters SHALL NOT affect

5115 the evaluation of an expression. Functions SHALL NOT have side effects, as evaluation order cannot be
5116 guaranteed in a standard way.

5117 **A.4 Functions, data types, attributes and algorithms planned for** 5118 **deprecation**

5119 The following functions, data types and algorithms have been defined by previous versions of XACML
5120 and newer and better alternatives are defined in XACML 3.0. Their use is discouraged for new use and
5121 they are candidates for deprecation in future versions of XACML.

5122 The following xpath based functions have been replaced with equivalent functions which use the new
5123 urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression datatype instead of strings.

- 5124 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-count
- 5125 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-count
- 5126 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal
- 5127 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-equal
- 5128 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-match
- 5129 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-match

5130 The following URI and string concatenation function has been replaced with a string to URI conversion
5131 function, which allows the use of the general string functions with URI through string conversion.

- 5132 • urn:oasis:names:tc:xacml:2.0:function:uri-string-concatenate
- 5133 • Replaced by urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI

5134 The following identifiers have been replaced with official identifiers defined by W3C.

- 5135 • <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration>
- 5136 • Replaced with <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- 5137 • <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration>
- 5138 • Replaced with <http://www.w3.org/2001/XMLSchema#yearMonthDuration>

5139 The following functions have been replaced with functions which use the updated dayTimeDuration and
5140 yearMonthDuration data types.

- 5141 • urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal
- 5142 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-equal
- 5143 • urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-equal
- 5144 • Replaced with urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-equal
- 5145 • urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration
- 5146 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-add-dayTimeDuration
- 5147 • urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration
- 5148 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-add-yearMonthDuration
- 5149 • urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration
- 5150 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-dayTimeDuration
- 5151 • urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-yearMonthDuration
- 5152 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-yearMonthDuration
- 5153 • urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration
- 5154 • Replaced with urn:oasis:names:tc:xacml:3.0:function:date-add-yearMonthDuration
- 5155 • urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration
- 5156 • Replaced with urn:oasis:names:tc:xacml:3.0:function:date-subtract-yearMonthDuration

- 5157 The following attribute identifiers have been replaced with new identifiers
- 5158 • urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address
- 5159 • Replaced with urn:oasis:names:tc:xacml:3.0:subject:authn-locality:ip-
- 5160 address
- 5161 • urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name
- 5162 • Replaced with urn:oasis:names:tc:xacml:3.0:subject:authn-
- 5163 locality:dns-name
- 5164
- 5165 The following combining algorithms have been replaced with new variants which allow for better handling
- 5166 of “Indeterminate” results.
- 5167 • urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides
- 5168 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides
- 5169 • urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides
- 5170 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides
- 5171 • urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides
- 5172 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides
- 5173 • urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides
- 5174 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides
- 5175 • urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides
- 5176 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides
- 5177 • urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides
- 5178 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-overrides
- 5179 • urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-overrides
- 5180 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-overrides
- 5181 • urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-overrides
- 5182 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-overrides

5183 Appendix B. XACML identifiers (normative)

5184 This section defines standard identifiers for commonly used entities.

5185 B.1 XACML namespaces

5186 XACML is defined using this identifier.

5187 `urn:oasis:names:tc:xacml:3.0:core:schema`

5188 B.2 Attribute categories

5189 The following **attribute** category identifiers MUST be used when an XACML 2.0 or earlier **policy** or
5190 request is translated into XACML 3.0.

5191 **Attributes** previously placed in the **Resource**, **Action**, and **Environment** sections of a request are
5192 placed in an **attribute** category with the following identifiers respectively. It is RECOMMENDED that they
5193 are used to list **attributes of resources**, **actions**, and the **environment** respectively when authoring
5194 XACML 3.0 **policies** or requests.

5195 `urn:oasis:names:tc:xacml:3.0:attribute-category:resource`

5196 `urn:oasis:names:tc:xacml:3.0:attribute-category:action`

5197 `urn:oasis:names:tc:xacml:3.0:attribute-category:environment`

5198 **Attributes** previously placed in the **Subject** section of a request are placed in an **attribute** category
5199 which is identical of the **subject** category in XACML 2.0, as defined below. It is RECOMMENDED that
5200 they are used to list **attributes of subjects** when authoring XACML 3.0 **policies** or requests.

5201 This identifier indicates the system entity that initiated the **access** request. That is, the initial entity in a
5202 request chain. If **subject** category is not specified in XACML 2.0, this is the default translation value.

5203 `urn:oasis:names:tc:xacml:1.0:subject-category:access-subject`

5204 This identifier indicates the system entity that will receive the results of the request (used when it is
5205 distinct from the access-**subject**).

5206 `urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject`

5207 This identifier indicates a system entity through which the **access** request was passed.

5208 `urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject`

5209 This identifier indicates a system entity associated with a local or remote codebase that generated the
5210 request. Corresponding **subject attributes** might include the URL from which it was loaded and/or the
5211 identity of the code-signer.

5212 `urn:oasis:names:tc:xacml:1.0:subject-category:codebase`

5213 This identifier indicates a system entity associated with the computer that initiated the **access** request.
5214 An example would be an IPsec identity.

5215 `urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine`

5216 B.3 Data-types

5217 The following identifiers indicate data-types that are defined in Section A.2.

5218 `urn:oasis:names:tc:xacml:1.0:data-type:x500Name`.

5219 `urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name`

5220 `urn:oasis:names:tc:xacml:2.0:data-type:ipAddress`

5221 `urn:oasis:names:tc:xacml:2.0:data-type:dnsName`

5222 `urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression`

- 5223 The following data-type identifiers are defined by XML Schema [XS].
- 5224 <http://www.w3.org/2001/XMLSchema#string>
- 5225 <http://www.w3.org/2001/XMLSchema#boolean>
- 5226 <http://www.w3.org/2001/XMLSchema#integer>
- 5227 <http://www.w3.org/2001/XMLSchema#double>
- 5228 <http://www.w3.org/2001/XMLSchema#time>
- 5229 <http://www.w3.org/2001/XMLSchema#date>
- 5230 <http://www.w3.org/2001/XMLSchema#dateTime>
- 5231 <http://www.w3.org/2001/XMLSchema#anyURI>
- 5232 <http://www.w3.org/2001/XMLSchema#hexBinary>
- 5233 <http://www.w3.org/2001/XMLSchema#base64Binary>
- 5234 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.
- 5235
- 5236 <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- 5237 <http://www.w3.org/2001/XMLSchema#yearMonthDuration>

5238 B.4 Subject attributes

- 5239 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).
- 5240
- 5241 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.
- 5242
- 5243 This identifier indicates the name of the **subject**.
- 5244 `urn:oasis:names:tc:xacml:1.0:subject:subject-id`
- 5245 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.
- 5246
- 5247 `urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier`
- 5248 This identifier indicates a public key used to confirm the **subject's** identity.
- 5249 `urn:oasis:names:tc:xacml:1.0:subject:key-info`
- 5250 This identifier indicates the time at which the **subject** was authenticated.
- 5251 `urn:oasis:names:tc:xacml:1.0:subject:authentication-time`
- 5252 This identifier indicates the method used to authenticate the **subject**.
- 5253 `urn:oasis:names:tc:xacml:1.0:subject:authentication-method`
- 5254 This identifier indicates the time at which the **subject** initiated the **access** request, according to the **PEP**.
- 5255 `urn:oasis:names:tc:xacml:1.0:subject:request-time`
- 5256 This identifier indicates the time at which the **subject's** current session began, according to the **PEP**.
- 5257 `urn:oasis:names:tc:xacml:1.0:subject:session-start-time`
- 5258 The following identifiers indicate the location where authentication credentials were activated.
- 5259 This identifier indicates that the location is expressed as an IP address.
- 5260 `urn:oasis:names:tc:xacml:3.0:subject:authn-locality:ip-address`
- 5261 The corresponding **attribute** SHALL be of data-type "`urn:oasis:names:tc:xacml:2.0:data-type:ipAddress`".
- 5262 This identifier indicates that the location is expressed as a DNS name.
- 5263 `urn:oasis:names:tc:xacml:3.0:subject:authn-locality:dns-name`
- 5264 The corresponding **attribute** SHALL be of data-type "`urn:oasis:names:tc:xacml:2.0:data-type:dnsName`".

5265 Where a suitable **attribute** is already defined in LDAP [LDAP-1], [LDAP-2], the XACML identifier SHALL
5266 be formed by adding the **attribute** name to the URI of the LDAP specification. For example, the **attribute**
5267 name for the userPassword defined in the RFC 2256 SHALL be:
5268 `http://www.ietf.org/rfc/rfc2256.txt#userPassword`

5269 B.5 Resource attributes

5270 These identifiers indicate **attributes** of the **resource**. When used, it is RECOMMENDED they appear
5271 within the <Attributes> element of the request **context** with Category
5272 `urn:oasis:names:tc:xacml:3.0:attribute-category:resource`.

5273 This **attribute** identifies the **resource** to which **access** is requested.

5274 `urn:oasis:names:tc:xacml:1.0:resource:resource-id`

5275 This **attribute** identifies the namespace of the top element(s) of the contents of the <Content> element.
5276 In the case where the **resource** content is supplied in the request **context** and the **resource**
5277 namespaces are defined in the **resource**, the **PEP** MAY provide this **attribute** in the request to indicate
5278 the namespaces of the **resource** content. In this case there SHALL be one value of this **attribute** for
5279 each unique namespace of the top level elements in the <Content> element. The type of the
5280 corresponding **attribute** SHALL be “`http://www.w3.org/2001/XMLSchema#anyURI`”.

5281 `urn:oasis:names:tc:xacml:2.0:resource:target-namespace`

5282 B.6 Action attributes

5283 These identifiers indicate **attributes** of the **action** being requested. When used, it is RECOMMENDED
5284 they appear within the <Attributes> element of the request **context** with Category
5285 `urn:oasis:names:tc:xacml:3.0:attribute-category:action`.

5286 This **attribute** identifies the **action** for which **access** is requested.

5287 `urn:oasis:names:tc:xacml:1.0:action:action-id`

5288 Where the **action** is implicit, the value of the action-id **attribute** SHALL be

5289 `urn:oasis:names:tc:xacml:1.0:action:implied-action`

5290 This **attribute** identifies the namespace in which the action-id **attribute** is defined.

5291 `urn:oasis:names:tc:xacml:1.0:action:action-namespace`

5292 B.7 Environment attributes

5293 These identifiers indicate **attributes** of the **environment** within which the **decision request** is to be
5294 evaluated. When used in the **decision request**, it is RECOMMENDED they appear in the
5295 <Attributes> element of the request **context** with Category `urn:oasis:names:tc:xacml:3.0:attribute-`
5296 `category:environment`.

5297 This identifier indicates the current time at the **context handler**. In practice it is the time at which the
5298 request **context** was created. For this reason, if these identifiers appear in multiple places within a
5299 <Policy> or <PolicySet>, then the same value SHALL be assigned to each occurrence in the
5300 evaluation procedure, regardless of how much time elapses between the processing of the occurrences.

5301 `urn:oasis:names:tc:xacml:1.0:environment:current-time`

5302 The corresponding **attribute** SHALL be of data-type “`http://www.w3.org/2001/XMLSchema#time`”.

5303 `urn:oasis:names:tc:xacml:1.0:environment:current-date`

5304 The corresponding **attribute** SHALL be of data-type “`http://www.w3.org/2001/XMLSchema#date`”.

5305 `urn:oasis:names:tc:xacml:1.0:environment:current-dateTime`

5306 The corresponding **attribute** SHALL be of data-type “`http://www.w3.org/2001/XMLSchema#dateTime`”.

5307 **B.8 Status codes**

5308 The following status code values are defined.

5309 This identifier indicates success.

5310 urn:oasis:names:tc:xacml:1.0:status:ok

5311 This identifier indicates that all the **attributes** necessary to make a **policy decision** were not available
5312 (see Section 5.58).

5313 urn:oasis:names:tc:xacml:1.0:status:missing-attribute

5314 This identifier indicates that some **attribute** value contained a syntax error, such as a letter in a numeric
5315 field.

5316 urn:oasis:names:tc:xacml:1.0:status:syntax-error

5317 This identifier indicates that an error occurred during **policy** evaluation. An example would be division by
5318 zero.

5319 urn:oasis:names:tc:xacml:1.0:status:processing-error

5320 **B.9 Combining algorithms**

5321 The deny-overrides **rule-combining algorithm** has the following value for the ruleCombiningAlgId
5322 attribute:

5323 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides

5324 The deny-overrides **policy-combining algorithm** has the following value for the
5325 policyCombiningAlgId attribute:

5326 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides

5327 The permit-overrides **rule-combining algorithm** has the following value for the ruleCombiningAlgId
5328 attribute:

5329 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides

5330 The permit-overrides **policy-combining algorithm** has the following value for the
5331 policyCombiningAlgId attribute:

5332 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides

5333 The first-applicable **rule-combining algorithm** has the following value for the ruleCombiningAlgId
5334 attribute:

5335 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable

5336 The first-applicable **policy-combining algorithm** has the following value for the
5337 policyCombiningAlgId attribute:

5338 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable

5339 The only-one-applicable-policy **policy-combining algorithm** has the following value for the
5340 policyCombiningAlgId attribute:

5341 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable

5342 The ordered-deny-overrides **rule-combining algorithm** has the following value for the
5343 ruleCombiningAlgId attribute:

5344 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides

5345 The ordered-deny-overrides **policy-combining algorithm** has the following value for the
5346 policyCombiningAlgId attribute:

5347 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-
5348 overrides

5349 The ordered-permit-overrides **rule-combining algorithm** has the following value for the
5350 ruleCombiningAlgId attribute:

5351 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-
5352 overrides

5353 The ordered-permit-overrides **policy-combining algorithm** has the following value for the
5354 policyCombiningAlgId attribute:

5355 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-
5356 overrides

5357 The deny-unless-permit **rule-combining algorithm** has the following value for the
5358 policyCombiningAlgId attribute:

5359 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit

5360 The permit-unless-deny **rule-combining algorithm** has the following value for the
5361 policyCombiningAlgId attribute:

5362 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-unless-deny

5363 The deny-unless-permit **policy-combining algorithm** has the following value for the
5364 policyCombiningAlgId attribute:

5365 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit

5366 The permit-unless-deny **policy-combining algorithm** has the following value for the
5367 policyCombiningAlgId attribute:

5368 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-unless-deny

5369 The legacy deny-overrides **rule-combining algorithm** has the following value for the
5370 ruleCombiningAlgId attribute:

5371 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides

5372 The legacy deny-overrides **policy-combining algorithm** has the following value for the
5373 policyCombiningAlgId attribute:

5374 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides

5375 The legacy permit-overrides **rule-combining algorithm** has the following value for the
5376 ruleCombiningAlgId attribute:

5377 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides

5378 The legacy permit-overrides **policy-combining algorithm** has the following value for the
5379 policyCombiningAlgId attribute:

5380 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides

5381 The legacy ordered-deny-overrides **rule-combining algorithm** has the following value for the
5382 ruleCombiningAlgId attribute:

5383 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides

5384 The legacy ordered-deny-overrides **policy-combining algorithm** has the following value for the
5385 policyCombiningAlgId attribute:

5386 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-
5387 overrides

5388 The legacy ordered-permit-overrides **rule-combining algorithm** has the following value for the
5389 ruleCombiningAlgId attribute:

5390 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-
5391 overrides

5392 The legacy ordered-permit-overrides **policy-combining algorithm** has the following value for the
5393 policyCombiningAlgId attribute:

5394 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-
5395 overrides

5396

5397

Appendix C. Combining algorithms (normative)

5398 This section contains a description of the **rule-** and **policy-combining algorithms** specified by XACML.
5399 Pseudo code is normative, descriptions in English are non-normative.

5400 The legacy **combining algorithms** are defined in previous versions of XACML, and are retained for
5401 compatibility reasons. It is RECOMMENDED that the new **combining algorithms** are used instead of the
5402 legacy **combining algorithms** for new use.

5403 Note that in each case an implementation is conformant as long as it produces the same result as is
5404 specified here, regardless of how and in what order the implementation behaves internally.

5405 C.1 Extended Indeterminate values

5406 Some combining algorithms are defined in terms of an extended set of "Indeterminate" values. See
5407 section 7.10 for the definition of the Extended Indeterminate values. For these algorithms, the **PDP** MUST
5408 keep track of the extended set of "Indeterminate" values during **rule** and **policy** combining.

5409 The output of a combining algorithm which does not track the extended set of "Indeterminate" values
5410 MUST be treated as "Indeterminate{DP}" for the value "Indeterminate" by a combining algorithm which
5411 tracks the extended set of "Indeterminate" values.

5412 A combining algorithm which does not track the extended set of "Indeterminate" values MUST treat the
5413 output of a combining algorithm which tracks the extended set of "Indeterminate" values as an
5414 "Indeterminate" for any of the possible values of the extended set of "Indeterminate".

5415 C.2 Deny-overrides

5416 This section defines the "Deny-overrides" **rule-combining algorithm** of a **policy** and **policy-combining**
5417 **algorithm** of a **policy set**.

5418 This **combining algorithm** makes use of the extended "Indeterminate".

5419 The **rule combining algorithm** defined here has the following identifier:

5420 `urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides`

5421 The **policy combining algorithm** defined here has the following identifier:

5422 `urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides`

5423 The following is a non-normative informative description of this **combining algorithm**.

5424 The deny overrides **combining algorithm** is intended for those cases where a deny
5425 decision should have priority over a permit decision. This algorithm has the following
5426 behavior.

- 5427 1. If any decision is "Deny", the result is "Deny".
5428 2. Otherwise, if any decision is "Indeterminate{DP}", the result is "Indeterminate{DP}".
5429 3. Otherwise, if any decision is "Indeterminate{D}" and another decision is "Indeterminate{P}" or
5430 Permit, the result is "Indeterminate{DP}".
5431 4. Otherwise, if any decision is "Indeterminate{D}", the result is "Indeterminate{D}".
5432 5. Otherwise, if any decision is "Permit", the result is "Permit".
5433 6. Otherwise, if any decision is "Indeterminate{P}", the result is "Indeterminate{P}".
5434 7. Otherwise, the result is "NotApplicable".

5435 The following pseudo-code represents the normative specification of this **combining algorithm**. The
5436 algorithm is presented here in a form where the input to it is an array with children (the **policies**, **policy**
5437 **sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set of
5438 obligations or advice provided by this algorithm is not deterministic.

```

5439 Decision denyOverridesCombiningAlgorithm(Node[] children)
5440 {
5441     Boolean atLeastOneErrorD = false;
5442     Boolean atLeastOneErrorP = false;
5443     Boolean atLeastOneErrorDP = false;
5444     Boolean atLeastOnePermit = false;
5445     for( i=0 ; i < lengthOf(children) ; i++ )
5446     {
5447         Decision decision = children[i].evaluate();
5448         if (decision == Deny)
5449         {
5450             return Deny;
5451         }
5452         if (decision == Permit)
5453         {
5454             atLeastOnePermit = true;
5455             continue;
5456         }
5457         if (decision == NotApplicable)
5458         {
5459             continue;
5460         }
5461         if (decision == Indeterminate{D})
5462         {
5463             atLeastOneErrorD = true;
5464             continue;
5465         }
5466         if (decision == Indeterminate{P})
5467         {
5468             atLeastOneErrorP = true;
5469             continue;
5470         }
5471         if (decision == Indeterminate{DP})
5472         {
5473             atLeastOneErrorDP = true;
5474             continue;
5475         }
5476     }
5477     if (atLeastOneErrorDP)
5478     {
5479         return Indeterminate{DP};
5480     }
5481     if (atLeastOneErrorD && (atLeastOneErrorP || atLeastOnePermit))
5482     {
5483         return Indeterminate{DP};
5484     }
5485     if (atLeastOneErrorD)
5486     {
5487         return Indeterminate{D};
5488     }
5489     if (atLeastOnePermit)
5490     {
5491         return Permit;
5492     }
5493     if (atLeastOneErrorP)
5494     {
5495         return Indeterminate{P};
5496     }
5497     return NotApplicable;
5498 }

```

5499 **Obligations** and **advice** shall be combined as described in Section 7.18.

5500 C.3 Ordered-deny-overrides

5501 The following specification defines the "Ordered-deny-overrides" **rule-combining algorithm** of a **policy**.

5502 The behavior of this algorithm is identical to that of the "Deny-overrides" **rule-combining**
5503 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL
5504 match the order as listed in the **policy**.

5505 The **rule combining algorithm** defined here has the following identifier:

5506 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides

5507 The following specification defines the "Ordered-deny-overrides" **policy-combining algorithm** of a
5508 **policy set**.

5509 The behavior of this algorithm is identical to that of the "Deny-overrides" **policy-combining**
5510 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL
5511 match the order as listed in the **policy set**.

5512 The **policy combining algorithm** defined here has the following identifier:

5513 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-
5514 overrides

5515 C.4 Permit-overrides

5516 This section defines the "Permit-overrides" **rule-combining algorithm** of a **policy** and **policy-combining**
5517 **algorithm** of a **policy set**.

5518 This **combining algorithm** makes use of the extended "Indeterminate".

5519 The **rule combining algorithm** defined here has the following identifier:

5520 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides

5521 The **policy combining algorithm** defined here has the following identifier:

5522 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides

5523 The following is a non-normative informative description of this combining algorithm.

5524 The permit overrides **combining algorithm** is intended for those cases where a permit
5525 decision should have priority over a deny decision. This algorithm has the following
5526 behavior.

- 5527 1. If any decision is "Permit", the result is "Permit".
- 5528 2. Otherwise, if any decision is "Indeterminate{DP}", the result is "Indeterminate{DP}".
- 5529 3. Otherwise, if any decision is "Indeterminate{P}" and another decision is
5530 "Indeterminate{D} or Deny, the result is "Indeterminate{DP}".
- 5531 4. Otherwise, if any decision is "Indeterminate{P}", the result is "Indeterminate{P}".
- 5532 5. Otherwise, if any decision is "Deny", the result is "Deny".
- 5533 6. Otherwise, if any decision is "Indeterminate{D}", the result is "Indeterminate{D}".
- 5534 7. Otherwise, the result is "NotApplicable".

5535 The following pseudo-code represents the normative specification of this **combining algorithm**. The
5536 algorithm is presented here in a form where the input to it is an array with all children (the **policies**, **policy**
5537 **sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set of
5538 obligations or advice provided by this algorithm is not deterministic.

```
5539 Decision permitOverridesCombiningAlgorithm(Node[] children)  
5540 {  
5541     Boolean atLeastOneErrorD = false;  
5542     Boolean atLeastOneErrorP = false;  
5543     Boolean atLeastOneErrorDP = false;  
5544     Boolean atLeastOneDeny = false;
```

```

5545 for( i=0 ; i < lengthOf(children) ; i++ )
5546 {
5547     Decision decision = children[i].evaluate();
5548     if (decision == Deny)
5549     {
5550         atLeastOneDeny = true;
5551         continue;
5552     }
5553     if (decision == Permit)
5554     {
5555         return Permit;
5556     }
5557     if (decision == NotApplicable)
5558     {
5559         continue;
5560     }
5561     if (decision == Indeterminate{D})
5562     {
5563         atLeastOneErrorD = true;
5564         continue;
5565     }
5566     if (decision == Indeterminate{P})
5567     {
5568         atLeastOneErrorP = true;
5569         continue;
5570     }
5571     if (decision == Indeterminate{DP})
5572     {
5573         atLeastOneErrorDP = true;
5574         continue;
5575     }
5576 }
5577 if (atLeastOneErrorDP)
5578 {
5579     return Indeterminate{DP};
5580 }
5581 if (atLeastOneErrorP && (atLeastOneErrorD || atLeastOneDeny))
5582 {
5583     return Indeterminate{DP};
5584 }
5585 if (atLeastOneErrorP)
5586 {
5587     return Indeterminate{P};
5588 }
5589 if (atLeastOneDeny)
5590 {
5591     return Deny;
5592 }
5593 if (atLeastOneErrorD)
5594 {
5595     return Indeterminate{D};
5596 }
5597 return NotApplicable;
5598 }

```

5599 **Obligations** and **advice** shall be combined as described in Section 7.18.

5600 C.5 Ordered-permit-overrides

5601 The following specification defines the "Ordered-permit-overrides" **rule-combining algorithm** of a **policy**.

5602 The behavior of this algorithm is identical to that of the "Permit-overrides" **rule-combining**
5603 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL
5604 match the order as listed in the **policy**.

5605 The **rule combining algorithm** defined here has the following identifier:
5606 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-
5607 overrides
5608 The following specification defines the "Ordered-permit-overrides" **policy-combining algorithm** of a
5609 **policy set**.
5610 The behavior of this algorithm is identical to that of the "Permit-overrides" **policy-combining**
5611 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL
5612 match the order as listed in the **policy set**.
5613 The **policy combining algorithm** defined here has the following identifier:
5614 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-
5615 overrides

5616 C.6 Deny-unless-permit

5617 This section defines the "Deny-unless-permit" **rule-combining algorithm** of a **policy** or **policy-**
5618 **combining algorithm** of a **policy set**.

5619 The **rule combining algorithm** defined here has the following identifier:
5620 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit

5621 The **policy combining algorithm** defined here has the following identifier:
5622 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit

5623 The following is a non-normative informative description of this **combining algorithm**.

5624 The "Deny-unless-permit" **combining algorithm** is intended for those cases where a
5625 permit decision should have priority over a deny decision, and an "Indeterminate" or
5626 "NotApplicable" must never be the result. It is particularly useful at the top level in a
5627 **policy** structure to ensure that a **PDP** will always return a definite "Permit" or "Deny"
5628 result. This algorithm has the following behavior.

- 5629 1. If any decision is "Permit", the result is "Permit".
- 5630 2. Otherwise, the result is "Deny".

5631 The following pseudo-code represents the normative specification of this **combining algorithm**. The
5632 algorithm is presented here in a form where the input to it is an array with all the children (the **policies**,
5633 **policy sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set
5634 of obligations or advice provided by this algorithm is not deterministic.

```
5635 Decision denyUnlessPermitCombiningAlgorithm(Node[] children)
5636 {
5637   for( i=0 ; i < lengthOf(children) ; i++ )
5638   {
5639     if (children[i].evaluate() == Permit)
5640     {
5641       return Permit;
5642     }
5643   }
5644   return Deny;
5645 }
```

5646 **Obligations** and **advice** shall be combined as described in Section 7.18.

5647 C.7 Permit-unless-deny

5648 This section defines the "Permit-unless-deny" **rule-combining algorithm** of a **policy** or **policy-**
5649 **combining algorithm** of a **policy set**.

5650 The **rule combining algorithm** defined here has the following identifier:

5651 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-unless-deny

5652 The **policy combining algorithm** defined here has the following identifier:
5653 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-unless-deny
5654 The following is a non-normative informative description of this **combining algorithm**.

5655 The "Permit-unless-deny" **combining algorithm** is intended for those cases where a
5656 deny decision should have priority over a permit decision, and an "Indeterminate" or
5657 "NotApplicable" must never be the result. It is particularly useful at the top level in a
5658 **policy** structure to ensure that a **PDP** will always return a definite "Permit" or "Deny"
5659 result. This algorithm has the following behavior.

- 5660 1. If any decision is "Deny", the result is "Deny".
- 5661 2. Otherwise, the result is "Permit".

5662 The following pseudo-code represents the normative specification of this **combining algorithm**. The
5663 algorithm is presented here in a form where the input to it is an array with all the children (the **policies**,
5664 **policy sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set
5665 of obligations or advice provided by this algorithm is not deterministic.

```
5666 Decision permitUnlessDenyCombiningAlgorithm(Node[] children)
5667 {
5668     for( i=0 ; i < lengthOf(children) ; i++ )
5669     {
5670         if (children[i].evaluate() == Deny)
5671         {
5672             return Deny;
5673         }
5674     }
5675     return Permit;
5676 }
```

5677 **Obligations** and **advice** shall be combined as described in Section 7.18.

5678 C.8 First-applicable

5679 This section defines the "First-applicable" **rule-combining algorithm** of a **policy** and **policy-combining**
5680 **algorithm** of a **policy set**.

5681 The **rule combining algorithm** defined here has the following identifier:

5682 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable

5683 The following is a non-normative informative description of the "First-Applicable" **rule-combining**
5684 **algorithm** of a **policy**.

5685 Each **rule** SHALL be evaluated in the order in which it is listed in the **policy**. For a particular
5686 **rule**, if the **target** matches and the **condition** evaluates to "True", then the evaluation of the
5687 **policy** SHALL halt and the corresponding **effect** of the **rule** SHALL be the result of the evaluation
5688 of the **policy** (i.e. "Permit" or "Deny"). For a particular **rule** selected in the evaluation, if the
5689 **target** evaluates to "False" or the **condition** evaluates to "False", then the next **rule** in the order
5690 SHALL be evaluated. If no further **rule** in the order exists, then the **policy** SHALL evaluate to
5691 "NotApplicable".

5692 If an error occurs while evaluating the **target** or **condition** of a **rule**, then the evaluation SHALL
5693 halt, and the **policy** shall evaluate to "Indeterminate", with the appropriate error status.

5694 The following pseudo-code represents the normative specification of this **rule-combining algorithm**.

```
5695 Decision firstApplicableEffectRuleCombiningAlgorithm(Rule[] rules)
5696 {
5697     for( i = 0 ; i < lengthOf(rules) ; i++ )
5698     {
5699         Decision decision = evaluate(rules[i]);
5700         if (decision == Deny)
5701         {
```

```

5702         return Deny;
5703     }
5704     if (decision == Permit)
5705     {
5706         return Permit;
5707     }
5708     if (decision == NotApplicable)
5709     {
5710         continue;
5711     }
5712     if (decision == Indeterminate)
5713     {
5714         return Indeterminate;
5715     }
5716 }
5717 return NotApplicable;
5718 }

```

5719 The **policy combining algorithm** defined here has the following identifier:

5720 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable

5721 The following is a non-normative informative description of the "First-applicable" **policy-combining**
5722 **algorithm** of a **policy set**.

5723 Each **policy** is evaluated in the order that it appears in the **policy set**. For a particular **policy**, if
5724 the **target** evaluates to "True" and the **policy** evaluates to a determinate value of "Permit" or
5725 "Deny", then the evaluation SHALL halt and the **policy set** SHALL evaluate to the **effect** value of
5726 that **policy**. For a particular **policy**, if the **target** evaluate to "False", or the **policy** evaluates to
5727 "NotApplicable", then the next **policy** in the order SHALL be evaluated. If no further **policy** exists
5728 in the order, then the **policy set** SHALL evaluate to "NotApplicable".

5729 If an error were to occur when evaluating the **target**, or when evaluating a specific **policy**, the
5730 reference to the **policy** is considered invalid, or the **policy** itself evaluates to "Indeterminate",
5731 then the evaluation of the **policy-combining algorithm** shall halt, and the **policy set** shall
5732 evaluate to "Indeterminate" with an appropriate error status.

5733 The following pseudo-code represents the normative specification of this policy-combination algorithm.

```

5734 Decision firstApplicableEffectPolicyCombiningAlgorithm(Policy[] policies)
5735 {
5736     for( i = 0 ; i < lengthOf(policies) ; i++ )
5737     {
5738         Decision decision = evaluate(policies[i]);
5739         if(decision == Deny)
5740         {
5741             return Deny;
5742         }
5743         if(decision == Permit)
5744         {
5745             return Permit;
5746         }
5747         if (decision == NotApplicable)
5748         {
5749             continue;
5750         }
5751         if (decision == Indeterminate)
5752         {
5753             return Indeterminate;
5754         }
5755     }
5756     return NotApplicable;
5757 }

```

5758 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

5759 C.9 Only-one-applicable

5760 This section defines the "Only-one-applicable" **policy-combining algorithm** of a **policy set**.

5761 The **policy combining algorithm** defined here has the following identifier:

5762 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable

5763 The following is a non-normative informative description of the "Only-one-applicable" **policy-combining algorithm** of a **policy set**.

5765 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".

5769 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**.

5771 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.

5774 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```
5775 Decision onlyOneApplicablePolicyPolicyCombiningAlogrithm(Policy[] policies)
5776 {
5777     Boolean          atLeastOne      = false;
5778     Policy           selectedPolicy = null;
5779     ApplicableResult appResult;
5780
5781     for ( i = 0; i < lengthOf(policies) ; i++ )
5782     {
5783         appResult = isApplicable(policies[I]);
5784
5785         if ( appResult == Indeterminate )
5786         {
5787             return Indeterminate;
5788         }
5789         if( appResult == Applicable )
5790         {
5791             if ( atLeastOne )
5792             {
5793                 return Indeterminate;
5794             }
5795             else
5796             {
5797                 atLeastOne      = true;
5798                 selectedPolicy = policies[i];
5799             }
5800         }
5801         if ( appResult == NotApplicable )
5802         {
5803             continue;
5804         }
5805     }
5806     if ( atLeastOne )
5807     {
5808         return evaluate(selectedPolicy);
5809     }
5810     else
5811     {
5812         return NotApplicable;
5813     }
5814 }
```

5815 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

5816 C.10 Legacy Deny-overrides

5817 This section defines the legacy “Deny-overrides” *rule-combining algorithm* of a *policy* and *policy-*
5818 *combining algorithm* of a *policy set*.

5819

5820 The *rule combining algorithm* defined here has the following identifier:

5821 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides

5822 The following is a non-normative informative description of this combining algorithm.

5823 The “Deny-overrides” rule combining algorithm is intended for those cases where a deny
5824 decision should have priority over a permit decision. This algorithm has the following
5825 behavior.

- 5826 1. If any rule evaluates to "Deny", the result is "Deny".
- 5827 2. Otherwise, if any rule having Effect="Deny" evaluates to "Indeterminate", the result is
5828 "Indeterminate".
- 5829 3. Otherwise, if any rule evaluates to "Permit", the result is "Permit".
- 5830 4. Otherwise, if any rule having Effect="Permit" evaluates to "Indeterminate", the result is
5831 "Indeterminate".
- 5832 5. Otherwise, the result is "NotApplicable".

5833 The following pseudo-code represents the normative specification of this *rule-combining algorithm*.

```
5834 Decision denyOverridesRuleCombiningAlgorithm(Rule[] rules)
5835 {
5836     Boolean atLeastOneError = false;
5837     Boolean potentialDeny = false;
5838     Boolean atLeastOnePermit = false;
5839     for( i=0 ; i < lengthOf(rules) ; i++ )
5840     {
5841         Decision decision = evaluate(rules[i]);
5842         if (decision == Deny)
5843         {
5844             return Deny;
5845         }
5846         if (decision == Permit)
5847         {
5848             atLeastOnePermit = true;
5849             continue;
5850         }
5851         if (decision == NotApplicable)
5852         {
5853             continue;
5854         }
5855         if (decision == Indeterminate)
5856         {
5857             atLeastOneError = true;
5858
5859             if (effect(rules[i]) == Deny)
5860             {
5861                 potentialDeny = true;
5862             }
5863             continue;
5864         }
5865     }
5866     if (potentialDeny)
5867     {
5868         return Indeterminate;
5869     }
5870     if (atLeastOnePermit)
5871     {
```

```

5872     return Permit;
5873 }
5874 if (atLeastOneError)
5875 {
5876     return Indeterminate;
5877 }
5878 return NotApplicable;
5879 }

```

5880 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

5881 The **policy combining algorithm** defined here has the following identifier:

5882 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides

5883 The following is a non-normative informative description of this combining algorithm.

5884 The "Deny-overrides" policy combining algorithm is intended for those cases where a
5885 deny decision should have priority over a permit decision. This algorithm has the
5886 following behavior.

- 5887 1. If any policy evaluates to "Deny", the result is "Deny".
- 5888 2. Otherwise, if any policy evaluates to "Indeterminate", the result is "Deny".
- 5889 3. Otherwise, if any policy evaluates to "Permit", the result is "Permit".
- 5890 4. Otherwise, the result is "NotApplicable".

5891 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```

5892 Decision denyOverridesPolicyCombiningAlgorithm(Policy[] policies)
5893 {
5894     Boolean atLeastOnePermit = false;
5895     for( i=0 ; i < lengthOf(policies) ; i++ )
5896     {
5897         Decision decision = evaluate(policies[i]);
5898         if (decision == Deny)
5899         {
5900             return Deny;
5901         }
5902         if (decision == Permit)
5903         {
5904             atLeastOnePermit = true;
5905             continue;
5906         }
5907         if (decision == NotApplicable)
5908         {
5909             continue;
5910         }
5911         if (decision == Indeterminate)
5912         {
5913             return Deny;
5914         }
5915     }
5916     if (atLeastOnePermit)
5917     {
5918         return Permit;
5919     }
5920     return NotApplicable;
5921 }

```

5922 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

5923 C.11 Legacy Ordered-deny-overrides

5924 The following specification defines the legacy "Ordered-deny-overrides" **rule-combining algorithm** of a
5925 **policy**.

5926 The behavior of this algorithm is identical to that of the “Deny-overrides” **rule-combining**
5927 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL
5928 match the order as listed in the **policy**.

5929 The **rule combining algorithm** defined here has the following identifier:

5930 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides

5931 The following specification defines the legacy “Ordered-deny-overrides” **policy-combining algorithm** of
5932 a **policy set**.

5933 The behavior of this algorithm is identical to that of the “Deny-overrides” **policy-combining**
5934 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL
5935 match the order as listed in the **policy set**.

5936 The **rule combining algorithm** defined here has the following identifier:

5937 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-
5938 overrides

5939 C.12 Legacy Permit-overrides

5940 This section defines the legacy “Permit-overrides” **rule-combining algorithm** of a **policy** and **policy-**
5941 **combining algorithm** of a **policy set**.

5942 The **rule combining algorithm** defined here has the following identifier:

5943 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides

5944 The following is a non-normative informative description of this combining algorithm.

5945 The “Permit-overrides” rule combining algorithm is intended for those cases where a
5946 permit decision should have priority over a deny decision. This algorithm has the
5947 following behavior.

- 5948 1. If any rule evaluates to "Permit", the result is "Permit".
- 5949 2. Otherwise, if any rule having Effect="Permit" evaluates to "Indeterminate" the result is
5950 "Indeterminate".
- 5951 3. Otherwise, if any rule evaluates to "Deny", the result is "Deny".
- 5952 4. Otherwise, if any rule having Effect="Deny" evaluates to "Indeterminate", the result is
5953 "Indeterminate".
- 5954 5. Otherwise, the result is "NotApplicable".

5955 The following pseudo-code represents the normative specification of this **rule-combining algorithm**.

```
5956 Decision permitOverridesRuleCombiningAlgorithm(Rule[] rules)
5957 {
5958     Boolean atLeastOneError = false;
5959     Boolean potentialPermit = false;
5960     Boolean atLeastOneDeny = false;
5961     for( i=0 ; i < lengthOf(rules) ; i++ )
5962     {
5963         Decision decision = evaluate(rules[i]);
5964         if (decision == Deny)
5965         {
5966             atLeastOneDeny = true;
5967             continue;
5968         }
5969         if (decision == Permit)
5970         {
5971             return Permit;
5972         }
5973         if (decision == NotApplicable)
5974         {
5975             continue;
5976         }
5977     }
5978 }
```

```

5977     if (decision == Indeterminate)
5978     {
5979         atLeastOneError = true;
5980
5981         if (effect(rules[i]) == Permit)
5982         {
5983             potentialPermit = true;
5984         }
5985         continue;
5986     }
5987 }
5988 if (potentialPermit)
5989 {
5990     return Indeterminate;
5991 }
5992 if (atLeastOneDeny)
5993 {
5994     return Deny;
5995 }
5996 if (atLeastOneError)
5997 {
5998     return Indeterminate;
5999 }
6000 return NotApplicable;
6001 }

```

6002 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

6003 The **policy combining algorithm** defined here has the following identifier:

6004 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides

6005 The following is a non-normative informative description of this combining algorithm.

6006 The "Permit-overrides" policy combining algorithm is intended for those cases where a
6007 permit decision should have priority over a deny decision. This algorithm has the
6008 following behavior.

- 6009 1. If any policy evaluates to "Permit", the result is "Permit".
- 6010 2. Otherwise, if any policy evaluates to "Deny", the result is "Deny".
- 6011 3. Otherwise, if any policy evaluates to "Indeterminate", the result is "Indeterminate".
- 6012 4. Otherwise, the result is "NotApplicable".

6013 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```

6014 Decision permitOverridesPolicyCombiningAlgorithm(Policy[] policies)
6015 {
6016     Boolean atLeastOneError = false;
6017     Boolean atLeastOneDeny = false;
6018     for( i=0 ; i < lengthOf(policies) ; i++ )
6019     {
6020         Decision decision = evaluate(policies[i]);
6021         if (decision == Deny)
6022         {
6023             atLeastOneDeny = true;
6024             continue;
6025         }
6026         if (decision == Permit)
6027         {
6028             return Permit;
6029         }
6030         if (decision == NotApplicable)
6031         {
6032             continue;
6033         }
6034         if (decision == Indeterminate)

```

```

6035     {
6036         atLeastOneError = true;
6037         continue;
6038     }
6039 }
6040 if (atLeastOneDeny)
6041 {
6042     return Deny;
6043 }
6044 if (atLeastOneError)
6045 {
6046     return Indeterminate;
6047 }
6048 return NotApplicable;
6049 }

```

6050 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

6051 **C.13 Legacy Ordered-permit-overrides**

6052 The following specification defines the legacy "Ordered-permit-overrides" **rule-combining algorithm** of a
6053 **policy**.

6054 The behavior of this algorithm is identical to that of the "Permit-overrides" **rule-combining**
6055 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL
6056 match the order as listed in the **policy**.

6057 The **rule combining algorithm** defined here has the following identifier:

6058 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-
6059 overrides

6060 The following specification defines the legacy "Ordered-permit-overrides" **policy-combining algorithm** of
6061 a **policy set**.

6062 The behavior of this algorithm is identical to that of the "Permit-overrides" **policy-combining**
6063 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL
6064 match the order as listed in the **policy set**.

6065 The **policy combining algorithm** defined here has the following identifier:

6066 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-
6067 overrides

6068

6069 Appendix D. Acknowledgements

6070 The following individuals have participated in the creation of this specification and are gratefully
6071 acknowledged:

6072

6073 Anil Saldhana

6074 Anil Tappetla

6075 Anne Anderson

6076 Anthony Nadalin

6077 Bill Parducci

6078 Craig Forster

6079 David Chadwick

6080 David Staggs

6081 Dilli Arumugam

6082 Duane DeCouteau

6083 Erik Rissanen

6084 Gareth Richards

6085 Hal Lockhart

6086 Jan Herrmann

6087 John Tolbert

6088 Ludwig Seitz

6089 Michiharu Kudo

6090 Naomaru Itoi

6091 Paul Tyson

6092 Prateek Mishra

6093 Rich Levinson

6094 Ronald Jacobson

6095 Seth Proctor

6096 Sridhar Muppidi

6097 Tim Moses

6098 Vernon Murdoch

6099 Cyril Dangerville (not a member of XACML TC)

6100

Appendix E. Revision History

6101

6102

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 <Content> element the context node of all xpath expressions and introduced categorization of XPath expressions so they point to a specific <Content> element.</p> <p>Added <Content> element to the policy issuer.</p> <p>Added description of the <PolicyIssuer> element.</p> <p>Updated the schema figure in the introduction to reflect the new AllOf/AnyOf schema.</p> <p>Remove duplicate <CombinerParameters> element in the <Policy> element in the schema.</p> <p>Removed default attributes in the schema. (Version in <Policy(Set)> and MustBePresent in <AttributeDesignator> in <AttributeSelector>)</p> <p>Removed references in section 7.3 to the <Condition> 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 <Attribute>. Also added this attribute in the associated schema file.</p> <p>Removed description of non-existing XML attribute "ResourceId" from the element <Result>.</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 <Description> to <Apply>.</p> <p>Added XPath versioning to the request.</p> <p>Added security considerations about denial service and the access-permitted function.</p> <p>Changed <Target> 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 <Function> 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 <Advice> feature</p> <p>Made obligations mandatory (per the 19th 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 <Request></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 <AttributeAssignment></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"> • uri-starts-with to anyURI-starts-with • uri-ends-with to anyURI-ends-with • uri-contains to anyURI-contains • uri-substring to anyURI-substring <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 <Policy(Set)IdReference> in <PolicyIdList>.</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 <Content> 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 <AttributeDesignator></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>

6103