



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

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

11 May 2017

Specification URIs

This version:

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

<http://docs.oasis-open.org/xacml/3.0/errata01/csprd02/xacml-3.0-core-spec-errata01-csprd02-redlined.html>

<http://docs.oasis-open.org/xacml/3.0/errata01/csprd02/xacml-3.0-core-spec-errata01-csprd02-redlined.pdf>

Previous version:

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

<http://docs.oasis-open.org/xacml/3.0/errata01/csprd01/xacml-3.0-core-spec-errata01-csprd01-redlined.html>

<http://docs.oasis-open.org/xacml/3.0/errata01/csprd01/xacml-3.0-core-spec-errata01-csprd01-redlined.pdf>

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@axiomantics.com), Axiomatics AB

Additional artifacts:

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

- List of Errata: *eXtensible Access Control Markup Language (XACML) Version 3.0 Errata 01*. Committee Specification Draft 02 / Public Review Draft 02. <http://docs.oasis-open.org/xacml/3.0/errata01/csprd02/xacml-3.0-core-spec-errata01-csprd02.html>.
- *eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01*. Edited by Erik Rissanen. 11 May 2017. OASIS Standard incorporating Public Review Draft 02 of Errata 01. <http://docs.oasis-open.org/xacml/3.0/errata01/csprd02/xacml-3.0-core-spec-errata01-csprd02-complete.html>.

- XML schema – unmodified from OASIS Standard: <http://docs.oasis-open.org/xacml/3.0/errata01/csprd02/schema/xacml-core-v3-schema-wd-17.xsd>.

Related work:

This specification provides Errata for:

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

Declared XML namespace:

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

Abstract:

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

Status:

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

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-redlined]

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

Edited by Erik Rissanen. 11 May 2017. OASIS Standard incorporating Public Review Draft 02 of Errata 01. <http://docs.oasis-open.org/xacml/3.0/errata01/csprd02/xacml-3.0-core-spec-errata01-csprd02-redlined.html>. Latest version: <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html>.

Notices

Copyright © OASIS Open 2017. All Rights Reserved.

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

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

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

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

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

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

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

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

Table of Contents

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

5.4 Element <PolicySetDefaults>	45
5.5 Element <XPathVersion>	46
5.6 Element <Target>	46
5.7 Element <AnyOf>	46
5.8 Element <AllOf>	47
5.9 Element <Match>	47
5.10 Element <PolicySetIdReference>	48
5.11 Element <PolicyIdReference>	48
5.12 Simple type VersionType	48
5.13 Simple type VersionMatchType	49
5.14 Element <Policy>	49
5.15 Element <PolicyDefaults>	51
5.16 Element <CombinerParameters>	51
5.17 Element <CombinerParameter>	52
5.18 Element <RuleCombinerParameters>	52
5.19 Element <PolicyCombinerParameters>	53
5.20 Element <PolicySetCombinerParameters>	53
5.21 Element <Rule>	54
5.22 Simple type EffectType	55
5.23 Element <VariableDefinition>	55
5.24 Element <VariableReference>	56
5.25 Element <Expression>	56
5.26 Element <Condition>	56
5.27 Element <Apply>	57
5.28 Element <Function>	57
5.29 Element <AttributeDesignator>	58
5.30 Element <AttributeSelector>	59
5.31 Element <AttributeValue>	60
5.32 Element <Obligations>	60
5.33 Element <AssociatedAdvice>	60
5.34 Element <Obligation>	61
5.35 Element <Advice>	61
5.36 Element <AttributeAssignment>	61
5.37 Element <ObligationExpressions>	62
5.38 Element <AdviceExpressions>	62
5.39 Element <ObligationExpression>	63
5.40 Element <AdviceExpression>	63
5.41 Element <AttributeAssignmentExpression>	64
5.42 Element <Request>	65
5.43 Element <RequestDefaults>	66
5.44 Element <Attributes>	66
5.45 Element <Content>	67
5.46 Element <Attribute>	67
5.47 Element <Response>	68
5.48 Element <Result>	68

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

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

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

1 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 ~~The set~~An element of **decision requests**, identified by definitions for an XACML **rule**, **policy**, or
93 **policy set** which matches specified values of resource, subject and **environment**, action that a,
94 or other custom attributes against those provided in the request context as a part of the process
95 of determining whether the rule, policy, or policy set is intended applicable to evaluate the current
96 decision.

97 Type Unification

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

105 1.1.2 Related terms

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

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

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

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

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

112 1.2 Terminology

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

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

118

119 Listings of XACML schema appear like this.

120

121 Example code listings appear like this.

122

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

- 126 • The prefix `xacml:` stands for the XACML 3.0 namespace.

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

135 1.3 Schema organization and namespaces

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

138 1.4 Normative References

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

174		Multimedia Environment, April 1993. Available at:
175		http://www.ifla.org/documents/infopol/copyright/perh2.txt
176	[RBAC]	David Ferraiolo and Richard Kuhn, <i>Role-Based Access Controls</i> , 15th National Computer Security Conference, 1992.
177		
178	[RFC2119]	S. Bradner, <i>Key words for use in RFCs to Indicate Requirement Levels</i> ,
179		http://www.ietf.org/rfc/rfc2119.txt , IETF RFC 2119, March 1997.
180	[RFC2396]	Berners-Lee T, Fielding R, Masinter L, <i>Uniform Resource Identifiers (URI):</i>
181		<i>Generic Syntax</i> . Available at: http://www.ietf.org/rfc/rfc2396.txt
182	[RFC2732]	Hinden R, Carpenter B, Masinter L, <i>Format for Literal IPv6 Addresses in URL's</i> .
183		Available at: http://www.ietf.org/rfc/rfc2732.txt
184	[RFC3198]	IETF RFC 3198: <i>Terminology for Policy-Based Management</i> , November 2001.
185		http://www.ietf.org/rfc/rfc3198.txt
186	[UAX15]	Mark Davis, Martin Dürst, <i>Unicode Standard Annex #15: Unicode Normalization</i>
187		<i>Forms, Unicode 5.1</i> , available from http://unicode.org/reports/tr15/
188	[UTR36]	Davis, Mark, Suignard, Michel, <i>Unicode Technocal Report #36: Unicode Security</i>
189		<i>Considerations</i> . Available at http://www.unicode.org/reports/tr36/
190	[XACMLAdmin]	OASIS Committee Draft 03, <i>XACML v3.0 Administration and Delegation Profile</i>
191		<i>Version 1.0</i> . 11 March 2010. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-
192		administration-v1-spec-cd-03-en.doc
193	[XACMLv1.0]	OASIS Standard, <i>Extensible access control markup language (XACML) Version</i>
194		<i>1.0</i> . 18 February 2003. http://www.oasis-
195		open.org/committees/download.php/2406/oasis-xacml-1.0.pdf
196	[XACMLv1.1]	OASIS Committee Specification, <i>Extensible access control markup language</i>
197		<i>(XACML) Version 1.1</i> . 7 August 2003. http://www.oasis-
198		open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf
199	[XF]	<i>XQuery 1.0 and XPath 2.0 Functions and Operators</i> , W3C Recommendation 23
200		January 2007. Available at: http://www.w3.org/TR/2007/REC-xpath-functions-
201		20070123/
202	[XML]	Bray, Tim, et.al. eds, <i>Extensible Markup Language (XML) 1.0 (Fifth Edition)</i> ,
203		W3C Recommendation 26 November 2008, available at
204		http://www.w3.org/TR/2008/REC-xml-20081126/
205	[XMLid]	Marsh, Jonathan, et.al. eds, <i>xml:id Version 1.0</i> . W3C Recommendation 9
206		September 2005. Available at: http://www.w3.org/TR/2005/REC-xml-id-
207		20050909/
208	[XS]	<i>XML Schema, parts 1 and 2</i> . Available at: http://www.w3.org/TR/xmlschema-1/
209		and http://www.w3.org/TR/xmlschema-2/
210	[XPath]	<i>XML Path Language (XPath), Version 1.0</i> , W3C Recommendation 16 November
211		1999. Available at: http://www.w3.org/TR/xpath
212	[XPathFunc]	<i>XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition)</i> , W3C
213		Recommendation 14 December 2010. Available at:
214		http://www.w3.org/TR/2010/REC-xpath-functions-20101214/
215	[XSLT]	<i>XSL Transformations (XSLT) Version 1.0</i> , W3C Recommendation 16 November
216		1999. Available at: http://www.w3.org/TR/xslt

217 1.5 Non-Normative References

218	[CM]	<i>Character model for the World Wide Web 1.0: Normalization</i> , W3C Working
219		Draft, 27 October 2005, http://www.w3.org/TR/2005/WD-charmod-norm-
220		20051027/ , World Wide Web Consortium.
221	[Hinton94]	Hinton, H, M, Lee, E, S, <i>The Compatibility of Policies</i> , Proceedings 2nd ACM
222		Conference on Computer and Communications Security, Nov 1994, Fairfax,
223		Virginia, USA.
224	[Sloman94]	Sloman, M. <i>Policy Driven Management for Distributed Systems</i> . Journal of
225		Network and Systems Management, Volume 2, part 4. Plenum Press. 1994.

226

2 Background (non-normative)

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

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

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

249 XML is a natural choice as the basis for the common security-policy language, due to the ease with which
250 its syntax and semantics can be extended to accommodate the unique requirements of this application,
251 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

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

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

277 2.2 Rule and policy combining

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

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

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

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

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

296 2.3 Combining algorithms

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

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

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

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

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

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

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

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

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

325 2.4 Multiple subjects

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

332 2.5 Policies based on subject and resource attributes

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

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

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

349 2.6 Multi-valued attributes

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

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

359 2.7 Policies based on resource contents

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

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

366 <Content> element of the **resource** to identify data in the information **resource** to be used in the **policy**
367 evaluation.
368 In cases where the information **resource** is not an XML document, specified **attributes** of the **resource**
369 can be referenced, as described in Section 2.5.

370 2.8 Operators

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

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

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

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

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

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

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

399 2.9 Policy distribution

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

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

411 2.10 Policy indexing

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

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

416 Two approaches are supported:

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

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

425 2.11 Abstraction layer

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

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

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

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

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

447 2.12 Actions performed in conjunction with enforcement

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

457 2.13 Supplemental information about a decision

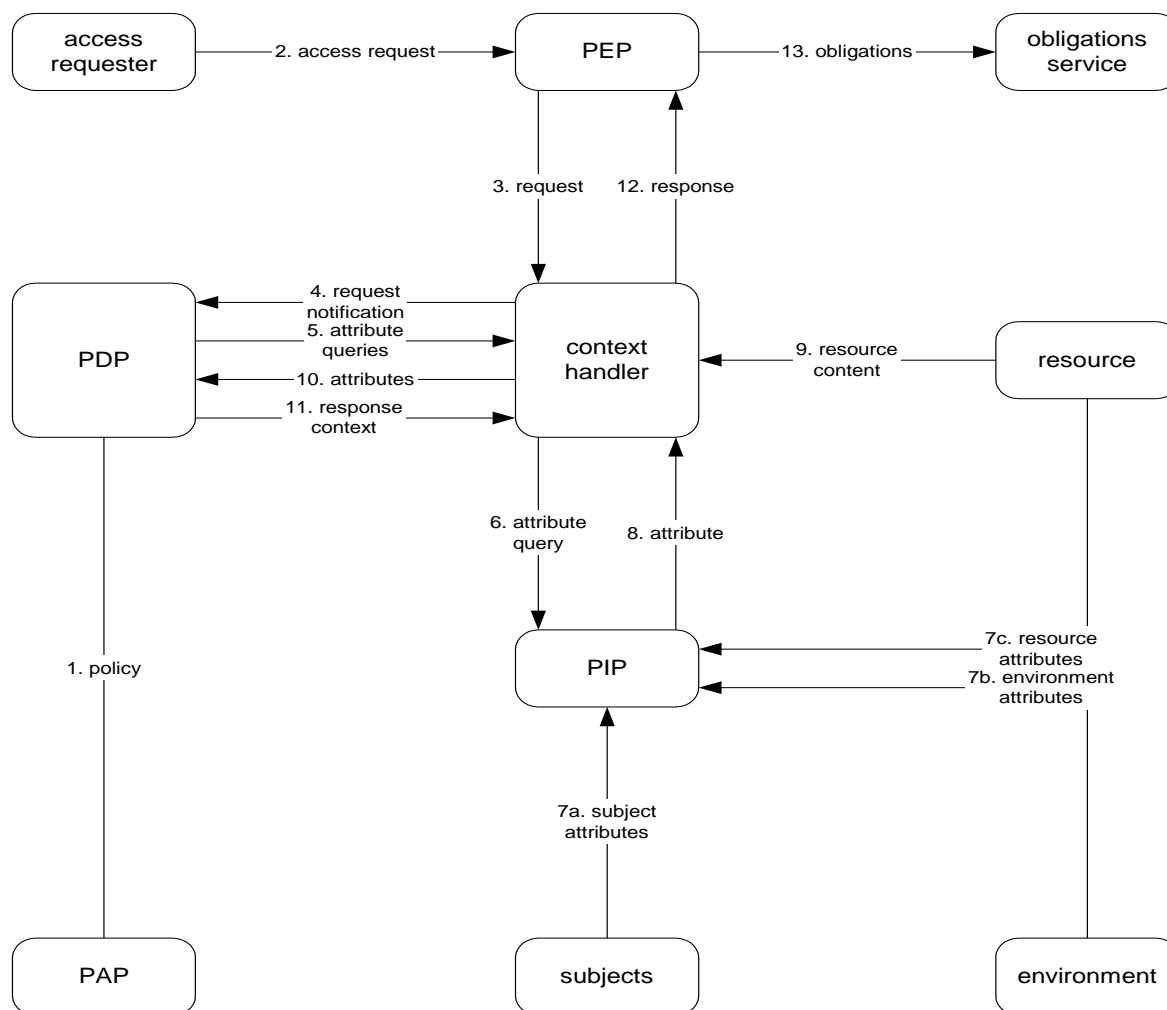
458 In some applications it is helpful to specify supplemental information about a decision. XACML provides
459 facilities to specify supplemental information about a decision with the <Advice> element. Such **advice**
460 may be safely ignored by the **PEP**.

461 3 Models (non-normative)

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

463 3.1 Data-flow model

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



465
466 Figure 1 - Data-flow diagram

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

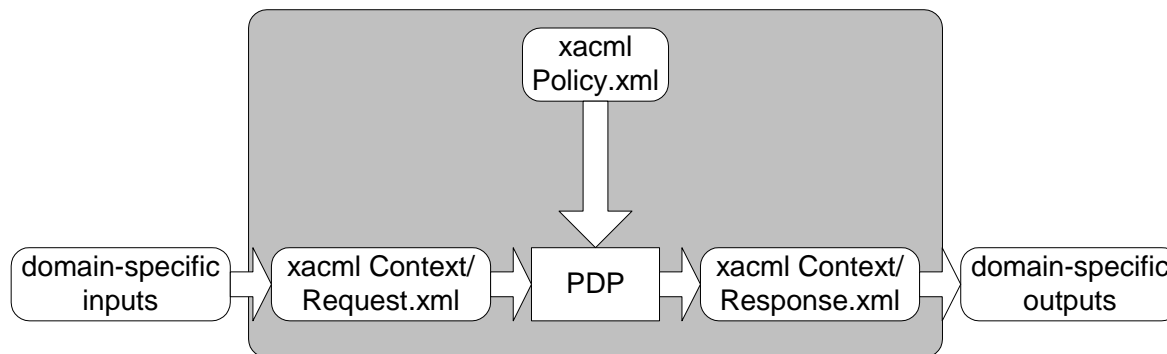
473 The model operates by the following steps.

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

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

497 3.2 XACML context

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



509
510 *Figure 2 - XACML context*

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

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

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

516 **3.3 Policy language model**

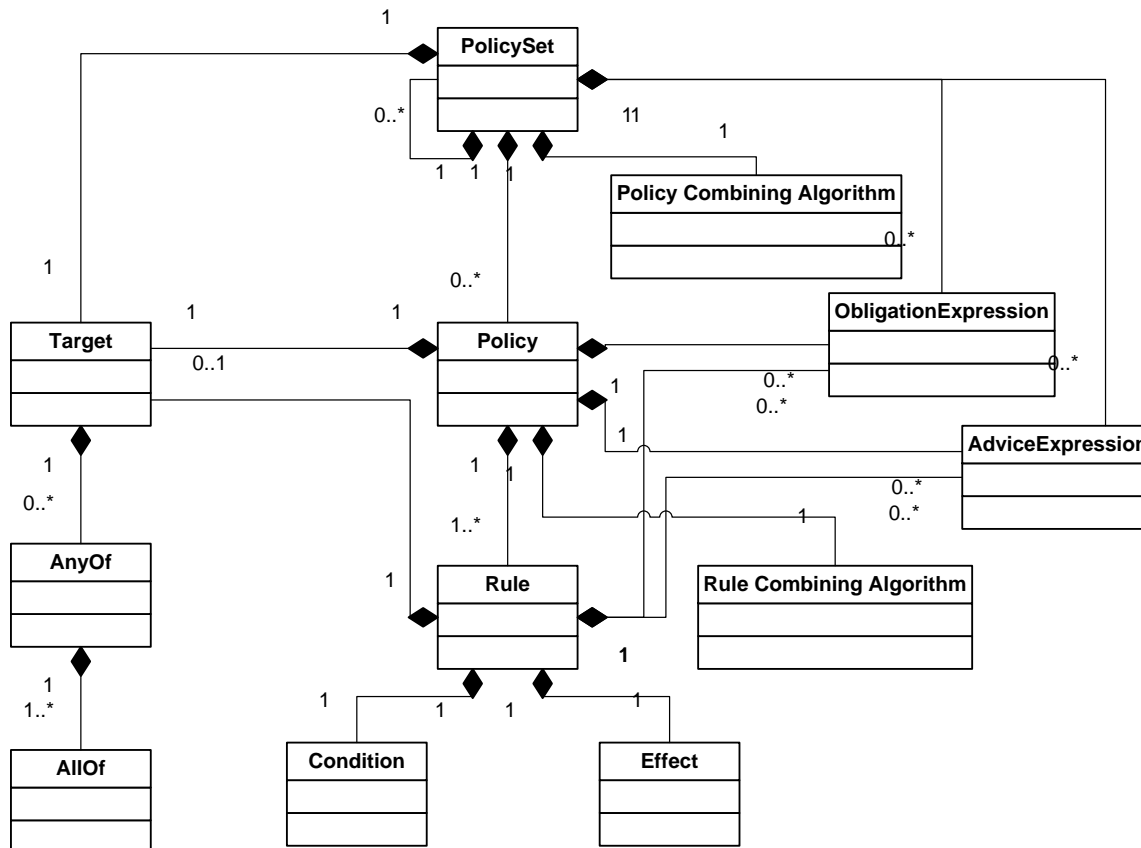
517 *The policy language model is shown in*

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

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

522 These are described in the following sub-sections.

523



524

525

526 *Figure 3 - Policy language model*

527 **3.3.1 Rule**

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

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

536 These are discussed in the following sub-sections.

537 3.3.1.1 Rule target

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

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

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

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

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

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

563 3.3.1.2 Effect

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

566 3.3.1.3 Condition

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

569 3.3.1.4 Obligation expressions

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

571 When a **PDP** evaluates a **rule** containing **obligation** expressions, it evaluates the **obligation** expressions
572 into **obligations** and returns certain of those **obligations** to the **PEP** in the response **context**. Section
573 7.18 explains which **obligations** are to be returned.

574 3.3.1.5 Advice

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

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

579 3.3.2 Policy

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

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

588 3.3.2.1 Policy target

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

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

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

608 3.3.2.2 Rule-combining algorithm

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

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

614 3.3.2.3 Obligation expressions

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

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

619 3.3.2.4 Advice

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

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

624 3.3.3 Policy set

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

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

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

633 3.3.3.1 Policy-combining algorithm

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

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

640 3.3.3.2 Obligation expressions

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

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

646 3.3.3.3 Advice expressions

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

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

652

4 Examples (non-normative)

653

This section contains two examples of the use of XACML for illustrative purposes. The first example is a relatively simple one to illustrate the use of **target**, **context**, matching functions and **subject attributes**.

654

655

The second example additionally illustrates the use of the **rule-combining algorithm**, **conditions** and **obligations**.

656

657

4.1 Example one

658

4.1.1 Example policy

659

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

660

661

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

662

663

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

664

665

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

705

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

706

707

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

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

751 4.1.2 Example request context

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

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

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

```

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

```

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

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

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

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

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

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

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

815 4.1.3 Example response context

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

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

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

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

833 [c6] closes the response *context*.

834 4.2 Example two

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

838 4.2.1 Example medical record instance

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

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

```

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

```

923 4.2.2 Example request context

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

```

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

```

```

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

```

1002 [e2] - [e4] Standard namespace declarations.

1003 [e6] - [e27] **Access subject attributes** are placed in the urn:oasis:names:tc:xacml:1.0:subject-
1004 category:access-subject **attribute** category of the <Request> element. Each **attribute** consists of the
1005 **attribute** meta-data and the **attribute** value. There is only one **subject** involved in this request. This
1006 value of the **attribute** category denotes the identity for which the request was issued.

1007 [e8] - [e13] **Subject** subject-id **attribute**.

1008 [e14] - [e20] **Subject** role **attribute**.

1009 [e21] - [e26] **Subject** physician-id **attribute**.

1010 [e28] - [e56] **Resource attributes** are placed in the urn:oasis:names:tc:xacml:3.0:attribute-
 1011 category:resource **attribute** category of the <Request> element. Each **attribute** consists of **attribute**
 1012 meta-data and an **attribute** value.

1013 [e30] - [e42] **Resource** content. The XML **resource** instance, **access** to all or part of which may be
 1014 requested, is placed here.

1015 [e43] - [e49] The identifier of the **Resource** instance for which **access** is requested, which is an XPath
 1016 expression into the <Content> element that selects the data to be accessed.

1017 [e57] - [e64] **Action attributes** are placed in the urn:oasis:names:tc:xacml:3.0:attribute-category:action
 1018 **attribute** category of the <Request> element.

1019 [e59] - [e63] **Action** identifier.

4.2.3 Example plain-language rules

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

- 1022 Rule 1: A person, identified by his or her patient number, may read any record for which he or she is
 1023 the designated patient.
- 1024 Rule 2: A person may read any record for which he or she is the designated parent or guardian, and
 1025 for which the patient is under 16 years of age.
- 1026 Rule 3: A physician may write to any medical element for which he or she is the designated primary
 1027 care physician, provided an email is sent to the patient.
- 1028 Rule 4: An administrator shall not be permitted to read or write to medical elements of a patient
 1029 record.

1030 These **rules** may be written by different **PAPs** operating independently, or by a single **PAP**.

4.2.4 Example XACML rule instances

4.2.4.1 Rule 1

1033 **Rule 1** illustrates a simple **rule** with a single <Condition> element. It also illustrates the use of the
 1034 <VariableDefinition> element to define a function that may be used throughout the **policy**. The
 1035 following XACML <Rule> instance expresses **Rule 1**:

```

1036 [f1] <?xml version="1.0" encoding="UTF-8"?>
1037 [f2] <Policy
1038 [f3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1039 [f4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1040 [f5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1041 [f6]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1042 [f7]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:1"
1043 [f8]   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1044 algorithm:deny-overrides"
1045 [f9]   Version="1.0">
1046 [f10]   <PolicyDefaults>
1047 [f11]     <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1048 [f12]   </PolicyDefaults>
1049 [f13]   <Target/>
1050 [f14]   <VariableDefinition VariableId="17590034">
1051 [f15]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1052 [f16]       <Apply
1053 [f17]         FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1054 [f18]           <AttributeDesignator
1055 [f19]             MustBePresent="false"
1056 [f20]             Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
1057 subject"
1058 [f21]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:patient-
1059 number"
1060 [f22]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1061 [f23]           </Apply>
1062 [f24]         <Apply
1063 [f25]           FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">

```

```

1064 [f26]         <AttributeSelector
1065 [f27]             MustBePresent="false"
1066 [f28]             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1067 [f29]             Path="md:record/md:patient/md:patient-number/text()"
1068 [f30]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1069 [f31]         </Apply>
1070 [f32]     </Apply>
1071 [f33] </VariableDefinition>
1072 [f34] <Rule
1073 [f35]     RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"
1074 [f36]     Effect="Permit">
1075 [f37]     <Description>
1076 [f38]         A person may read any medical record in the
1077 [f39]         http://www.med.example.com/schemas/record.xsd namespace
1078 [f40]         for which he or she is the designated patient
1079 [f41]     </Description>
1080 [f42]     <Target>
1081 [f43]         <AnyOf>
1082 [f44]             <AllOf>
1083 [f45]                 <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1084 [f46]                     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1085 [f47]                         >urn:example:med:schemas:record</AttributeValue>
1086 [f48]                     <AttributeDesignator
1087 [f49]                         MustBePresent="false"
1088 [f50]                         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1089 [f51]                         AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1090 [f52]                         DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1091 [f53]                 </Match>
1092 [f54]                 <Match
1093 [f55]                     MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1094 [f56]                         <AttributeValue
1095 [f57]                             DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1096 [f58]                             XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1097 [f59]                             >md:record</AttributeValue>
1098 [f60]                         <AttributeDesignator
1099 [f61]                             MustBePresent="false"
1100 [f62]                             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1101 [f63]                             AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1102 [f64]                             DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1103 [f65]                         </Match>
1104 [f66]                     </AllOf>
1105 [f67]                 </AnyOf>
1106 [f68]             <AnyOf>
1107 [f69]                 <AllOf>
1108 [f70]                     <Match
1109 [f71]                         MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1110 [f72]                             <AttributeValue
1111 [f73]                                 DataType="http://www.w3.org/2001/XMLSchema#string"
1112 [f74]                                 >read</AttributeValue>
1113 [f75]                             <AttributeDesignator
1114 [f76]                                 MustBePresent="false"
1115 [f77]                                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1116 [f78]                                 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1117 [f79]                                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1118 [f80]                             </Match>
1119 [f81]                         </AllOf>
1120 [f82]                     </AnyOf>
1121 [f83]                 </Target>
1122 [f84]             <Condition>
1123 [f85]                 <VariableReference VariableId="17590034"/>
1124 [f86]             </Condition>
1125 [f87]     </Rule>
1126 [f88] </Policy>

```

1127 [f3] - [f6] XML namespace declarations.

1128 [f11] XPath expressions in the **policy** are to be interpreted according to the 1.0 version of the XPath specification.

1130 [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**.

1132 [f15] The `FunctionId` attribute names the function to be used for comparison. In this case, comparison
1133 is done with the “urn:oasis:names:tc:xacml:1.0:function:string-equal” function; this function takes two
1134 arguments of type “http://www.w3.org/2001/XMLSchema#string”.

1135 [f17] The first argument of the variable definition is a function specified by the `FunctionId` attribute.
1136 Since urn:oasis:names:tc:xacml:1.0:function:string-equal takes arguments of type
1137 “http://www.w3.org/2001/XMLSchema#string” and `AttributeDesignator` selects a **bag** of type
1138 “http://www.w3.org/2001/XMLSchema#string”, “urn:oasis:names:tc:xacml:1.0:function:string-one-and-
1139 only” is used. This function guarantees that its argument evaluates to a **bag** containing exactly one
1140 value.

1141 [f18] The `AttributeDesignator` selects a **bag** of values for the patient-number **subject attribute** in
1142 the request **context**.

1143 [f25] The second argument of the variable definition is a function specified by the `FunctionId` attribute.
1144 Since “urn:oasis:names:tc:xacml:1.0:function:string-equal” takes arguments of type
1145 “http://www.w3.org/2001/XMLSchema#string” and the `AttributeSelector` selects a **bag** of type
1146 “http://www.w3.org/2001/XMLSchema#string”, “urn:oasis:names:tc:xacml:1.0:function:string-one-and-
1147 only” is used. This function guarantees that its argument evaluates to a **bag** containing exactly one
1148 value.

1149 [f26] The `<AttributeSelector>` element selects a **bag** of values from the **resource** content using a
1150 free-form XPath expression. In this case, it selects the value of the patient-number in the **resource**.
1151 Note that the namespace prefixes in the XPath expression are resolved with the standard XML
1152 namespace declarations.

1153 [f35] **Rule** identifier.

1154 [f36] **Rule effect** declaration. When a **rule** evaluates to ‘True’ it emits the value of the `Effect` attribute.
1155 This value is then combined with the `Effect` values of other **rules** according to the **rule-combining**
1156 **algorithm**.

1157 [f37] - [f41] Free form description of the **rule**.

1158 [f42] - [f83] A **rule target** defines a set of **decision requests** that the **rule** is intended to evaluate.

1159 [f43] - [f67] The `<AnyOf>` element contains a **disjunctive sequence** of `<AllOf>` elements. In this
1160 example, there is just one.

1161 [f44] - [f66] The `<AllOf>` element encloses the **conjunctive sequence** of `Match` elements. In this
1162 example, there are two.

1163 [f45] - [f53] The first `<Match>` element compares its first and second child elements according to the
1164 matching function. A match is positive if the value of the first argument matches any of the values
1165 selected by the second argument. This match compares the **target** namespace of the requested
1166 document with the value of “urn:example:med:schemas:record”.

1167 [f45] The `MatchId` attribute names the matching function.

1168 [f46] - [f47] Literal **attribute** value to match.

1169 [f48] - [f52] The `<AttributeDesignator>` element selects the **target** namespace from the **resource**
1170 contained in the request **context**. The **attribute** name is specified by the `AttributeId`.

1171 [f54] - [f65] The second `<Match>` element. This match compares the results of two XPath expressions
1172 applied to the `<Content>` element of the **resource** category. The second XPath expression is the
1173 location path to the requested XML element and the first XPath expression is the literal value “md:record”.
1174 The “xpath-node-match” function evaluates to “True” if the requested XML element is below the
1175 “md:record” element.

1176 [f68] - [f82] The `<AnyOf>` element contains a **disjunctive sequence** of `<AllOf>` elements. In this case,
1177 there is just one `<AllOf>` element.

1178 [f69] - [f81] The `<AllOf>` element contains a **conjunctive sequence** of `<Match>` elements. In this case,
1179 there is just one `<Match>` element.

1180 [f70] - [f80] The <Match> element compares its first and second child elements according to the matching
1181 function. The match is positive if the value of the first argument matches any of the values selected by
1182 the second argument. In this case, the value of the action-id **action attribute** in the request **context** is
1183 compared with the literal value "read".

1184 [f84] - [f86] The <Condition> element. A **condition** must evaluate to "True" for the **rule** to be
1185 applicable. This **condition** contains a reference to a variable definition defined elsewhere in the **policy**.

1186 4.2.4.2 Rule 2

1187 **Rule 2** illustrates the use of a mathematical function, i.e. the <Apply> element with functionId
1188 "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration" to calculate the date of the patient's
1189 sixteenth birthday. It also illustrates the use of **predicate** expressions, with the functionId
1190 "urn:oasis:names:tc:xacml:1.0:function:and". This example has one function embedded in the
1191 <Condition> element and another one referenced in a <VariableDefinition> element.

```
1192 [g1] <?xml version="1.0" encoding="UTF-8"?>
1193 [g2] <Policy
1194 [g3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1195 [g4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1196 [g5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1197 [g6]   xmlns:xf="http://www.w3.org/2005/xpath-functions"
1198 [g7]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1199 [g8]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"
1200 [g9]   Version="1.0"
1201 [g10]  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1202      algorithm:deny-overrides">
1203 [g11]  <PolicyDefaults>
1204 [g12]    <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1205 [g13]  </PolicyDefaults>
1206 [g14]  <Target/>
1207 [g15]  <VariableDefinition VariableId="17590035">
1208 [g16]    <Apply
1209 [g17]      FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal">
1210 [g18]      <Apply
1211 [g19]        FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
1212 [g20]          <AttributeDesignator
1213 [g21]            MustBePresent="false"
1214 [g22]            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
1215 [g23]            AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"
1216 [g24]            DataType="http://www.w3.org/2001/XMLSchema#date"/>
1217 [g25]          </Apply>
1218 [g26]        <Apply
1219 [g27]          FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration">
1220 [g28]            <Apply
1221 [g29]              FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
1222 [g30]                <AttributeSelector
1223 [g31]                  MustBePresent="false"
1224 [g32]                  Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1225 [g33]                  Path="md:record/md:patient/md:patientDoB/text()"
1226 [g34]                  DataType="http://www.w3.org/2001/XMLSchema#date"/>
1227 [g35]                </Apply>
1228 [g36]              <AttributeValue
1229 [g37]                DataType="http://www.w3.org/2001/XMLSchema#yearMonthDuration"
1230 [g38]                >P16Y</AttributeValue>
1231 [g39]            </Apply>
1232 [g40]          </Apply>
1233 [g41]        </VariableDefinition>
1234 [g42]      <Rule
1235 [g43]        RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"
1236 [g44]        Effect="Permit">
1237 [g45]        <Description>
1238 [g46]          A person may read any medical record in the
1239 [g47]          http://www.med.example.com/records.xsd namespace
1240 [g48]          for which he or she is the designated parent or guardian,
1241 [g49]          and for which the patient is under 16 years of age
1242 [g50]        </Description>
1243 [g51]        <Target>
1244 [g52]          <AnyOf>
1245 [g53]            <AllOf>
```

```

1246 [g54] <Match
1247 [g55]   MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1248 [g56]   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1249 [g57]     >urn:example:med:schemas:record</AttributeValue>
1250 [g58]   <AttributeDesignator
1251 [g59]     MustBePresent="false"
1252 [g60]     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1253 [g61]     AttributeId= "urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1254 [g62]     DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1255 [g63] </Match>
1256 [g64] <Match
1257 [g65]   MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1258 [g66]   <AttributeValue
1259 [g67]     DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1260 [g68]     XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1261 [g69]     >md:record</AttributeValue>
1262 [g70]   <AttributeDesignator
1263 [g71]     MustBePresent="false"
1264 [g72]     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1265 [g73]     AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1266 [g74]     DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1267 [g75] </Match>
1268 [g76] </AllOf>
1269 [g77] </AnyOf>
1270 [g78] <AnyOf>
1271 [g79]   <AllOf>
1272 [g80]     <Match
1273 [g81]       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1274 [g82]       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1275 [g83]         >read</AttributeValue>
1276 [g84]       <AttributeDesignator
1277 [g85]         MustBePresent="false"
1278 [g86]         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1279 [g87]         AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1280 [g88]         DataType="http://www.w3.org/2001/XMLSchema#string"/>
1281 [g89]       </Match>
1282 [g90]     </AllOf>
1283 [g91]   </AnyOf>
1284 [g92] </Target>
1285 [g93] <Condition>
1286 [g94]   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
1287 [g95]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1288 [g96]       <Apply
1289 [g97]         FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1290 [g98]           <AttributeDesignator
1291 [g99]             MustBePresent="false"
1292 [g100]           Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1293 [g101]           AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:parent-
1294 guardian-id"
1295 [g102]           DataType="http://www.w3.org/2001/XMLSchema#string"/>
1296 [g103]         </Apply>
1297 [g104]       <Apply
1298 [g105]         FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1299 [g106]           <AttributeSelector
1300 [g107]             MustBePresent="false"
1301 [g108]             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1302 [g109]             Path="md:record/md:parentGuardian/md:parentGuardianId/text () "
1303 [g110]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1304 [g111]           </Apply>
1305 [g112]         </Apply>
1306 [g113]       <VariableReference VariableId="17590035"/>
1307 [g114]     </Apply>
1308 [g115]   </Condition>
1309 [g116] </Rule>
1310 [g117] </Policy>

```

1311 [g15] - [g41] The <VariableDefinition> element contains part of the **condition** (i.e. is the patient
1312 under 16 years of age?). The patient is under 16 years of age if the current date is less than the date
1313 computed by adding 16 to the patient's date of birth.

1314 [g16] - [g40] "urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal" is used to compare the two date
1315 arguments.

1316 [g18] - [g25] The first date argument uses “urn:oasis:names:tc:xacml:1.0:function:date-one-and-only” to
 1317 ensure that the **bag** of values selected by its argument contains exactly one value of type
 1318 “http://www.w3.org/2001/XMLSchema#date”.

1319 [g20] The current date is evaluated by selecting the “urn:oasis:names:tc:xacml:1.0:environment:current-
 1320 date” **environment attribute**.

1321 [g26] - [g39] The second date argument uses “urn:oasis:names:tc:xacml:1.0:function:date-add-
 1322 yearMonthDuration” to compute the date of the patient’s sixteenth birthday by adding 16 years to the
 1323 patient’s date of birth. The first of its arguments is of type “http://www.w3.org/2001/XMLSchema#date”
 1324 and the second is of type “http://www.w3.org/TR/2007/REC-xpath-functions-20070123/#dt-
 1325 yearMonthDuration”.

1326 [g30] The <AttributeSelector> element selects the patient’s date of birth by taking the XPath
 1327 expression over the **resource** content.

1328 [g36] - [g38] Year Month Duration of 16 years.

1329 [g51] - [g92] **Rule** declaration and **rule target**. See **Rule** 1 in Section 4.2.4.1 for the detailed explanation
 1330 of these elements.

1331 [g93] - [g115] The <Condition> element. The **condition** must evaluate to “True” for the **rule** to be
 1332 applicable. This **condition** evaluates the truth of the statement: the requestor is the designated parent or
 1333 guardian and the patient is under 16 years of age. It contains one embedded <Apply> element and one
 1334 referenced <VariableDefinition> element.

1335 [g94] The **condition** uses the “urn:oasis:names:tc:xacml:1.0:function:and” function. This is a Boolean
 1336 function that takes one or more Boolean arguments (2 in this case) and performs the logical “AND”
 1337 operation to compute the truth value of the expression.

1338 [g95] - [g112] The first part of the **condition** is evaluated (i.e. is the requestor the designated parent or
 1339 guardian?). The function is “urn:oasis:names:tc:xacml:1.0:function:string-equal” and it takes two
 1340 arguments of type “http://www.w3.org/2001/XMLSchema#string”.

1341 [g96] designates the first argument. Since “urn:oasis:names:tc:xacml:1.0:function:string-equal” takes
 1342 arguments of type “http://www.w3.org/2001/XMLSchema#string”,
 1343 “urn:oasis:names:tc:xacml:1.0:function:string-one-and-only” is used to ensure that the **subject attribute**
 1344 “urn:oasis:names:tc:xacml:3.0:example:attribute:parent-guardian-id” in the request **context** contains
 1345 exactly one value.

1346 [g98] designates the first argument. The value of the **subject attribute**
 1347 “urn:oasis:names:tc:xacml:3.0:example:attribute:parent-guardian-id” is selected from the request **context**
 1348 using the <AttributeDesignator> element.

1349 [g104] As above, the “urn:oasis:names:tc:xacml:1.0:function:string-one-and-only” is used to ensure that
 1350 the **bag** of values selected by its argument contains exactly one value of type
 1351 “http://www.w3.org/2001/XMLSchema#string”.

1352 [g106] The second argument selects the value of the <md:parentGuardianId> element from the
 1353 **resource** content using the <AttributeSelector> element. This element contains a free-form XPath
 1354 expression, pointing into the <Content> element of the resource category. Note that all namespace
 1355 prefixes in the XPath expression are resolved with standard namespace declarations. The
 1356 AttributeSelector evaluates to the **bag** of values of type
 1357 “http://www.w3.org/2001/XMLSchema#string”.

1358 [g113] references the <VariableDefinition> element, where the second part of the **condition** is
 1359 defined.

1360 4.2.4.3 Rule 3

1361 **Rule 3** illustrates the use of an **obligation** expression.

```

1362 [h1] <?xml version="1.0" encoding="UTF-8"?>
1363 [h2] <Policy
1364 [h3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1365 [h4]   xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1366 [h5]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

```

1367 [h6] xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
1368 http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
1369 [h7] xmlns:md="http://www.med.example.com/schemas/record.xsd"
1370 [h8] PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:3"
1371 [h9] Version="1.0"
1372 [h10] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1373 algorithm:deny-overrides">
1374 [h11] <Description>
1375 [h12] Policy for any medical record in the
1376 [h13] http://www.med.example.com/schemas/record.xsd namespace
1377 [h14] </Description>
1378 [h15] <PolicyDefaults>
1379 [h16] <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1380 [h17] </PolicyDefaults>
1381 [h18] <Target>
1382 [h19] <AnyOf>
1383 [h20] <AllOf>
1384 [h21] <Match
1385 [h22] MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1386 [h23] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1387 [h24] >urn:example:med:schemas:record</AttributeValue>
1388 [h25] <AttributeDesignator
1389 [h26] MustBePresent="false"
1390 [h27] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1391 [h28] AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1392 [h29] DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1393 [h30] </Match>
1394 [h31] </AllOf>
1395 [h32] </AnyOf>
1396 [h33] </Target>
1397 [h34] <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:3"
1398 [h35] Effect="Permit">
1399 [h36] <Description>
1400 [h37] A physician may write any medical element in a record
1401 [h38] for which he or she is the designated primary care
1402 [h39] physician, provided an email is sent to the patient
1403 [h40] </Description>
1404 [h41] <Target>
1405 [h42] <AnyOf>
1406 [h43] <AllOf>
1407 [h44] <Match
1408 [h45] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1409 [h46] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1410 [h47] >physician</AttributeValue>
1411 [h48] <AttributeDesignator
1412 [h49] MustBePresent="false"
1413 [h50] Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1414 [h51] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
1415 [h52] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1416 [h53] </Match>
1417 [h54] </AllOf>
1418 [h55] </AnyOf>
1419 [h56] <AnyOf>
1420 [h57] <AllOf>
1421 [h58] <Match
1422 [h59] MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1423 [h60] <AttributeValue
1424 [h61] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1425 [h62] XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1426 [h63] >md:record/md:medical</AttributeValue>
1427 [h64] <AttributeDesignator
1428 [h65] MustBePresent="false"
1429 [h66] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1430 [h67] AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1431 [h68] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1432 [h69] </Match>
1433 [h70] </AllOf>
1434 [h71] </AnyOf>
1435 [h72] <AnyOf>
1436 [h73] <AllOf>
1437 [h74] <Match
1438 [h75] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1439 [h76] <AttributeValue

```

```

1440 [h77]         DataType="http://www.w3.org/2001/XMLSchema#string"
1441 [h78]         >write</AttributeValue>
1442 [h79]         <AttributeDesignator
1443 [h80]             MustBePresent="false"
1444 [h81]             Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1445 [h82]             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1446 [h83]             DataType="http://www.w3.org/2001/XMLSchema#string"/>
1447 [h84]         </Match>
1448 [h85]     </AllOf>
1449 [h86] </AnyOf>
1450 [h87] </Target>
1451 [h88] <Condition>
1452 [h89]     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1453 [h90]         <Apply
1454 [h91]             FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1455 [h92]             <AttributeDesignator
1456 [h93]                 MustBePresent="false"
1457 [h94]                 Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1458 [h95]                 AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:physician-id"
1459 [h96]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1460 [h97]             </Apply>
1461 [h98]         <Apply
1462 [h99]             FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1463 [h100]             <AttributeSelector
1464 [h101]                 MustBePresent="false"
1465 [h102]                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1466 [h103]                 Path="md:record/md:primaryCarePhysician/md:registrationID/text()"
1467 [h104]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1468 [h105]             </Apply>
1469 [h106]         </Apply>
1470 [h107]     </Condition>
1471 [h108] </Rule>
1472 [h109] <ObligationExpressions>
1473 [h110]     <ObligationExpression
1474 [h111]         ObligationId="urn:oasis:names:tc:xacml:example:obligation:email"
1475 [h112]         FulfillOn="Permit">
1476 [h113]         <AttributeAssignmentExpression
1477 [h114]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:mailto">
1478 [h115]             <AttributeSelector
1479 [h116]                 MustBePresent="true"
1480 [h117]                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1481 [h118]                 Path="md:record/md:patient/md:patientContact/md:email"
1482 [h119]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1483 [h120]             </AttributeAssignmentExpression>
1484 [h121]         <AttributeAssignmentExpression
1485 [h122]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
1486 [h123]             <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1487 [h124]                 >Your medical record has been accessed by:</AttributeValue>
1488 [h125]             </AttributeAssignmentExpression>
1489 [h126]         <AttributeAssignmentExpression
1490 [h127]             AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:text">
1491 [h128]             <AttributeDesignator
1492 [h129]                 MustBePresent="false"
1493 [h130]                 Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1494 [h131]                 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
1495 [h132]                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
1496 [h133]             </AttributeAssignmentExpression>
1497 [h134]         </ObligationExpression>
1498 [h135]     </ObligationExpressions>
1499 [h136] </Policy>

```

1500 [h2] - [h10] The <Policy> element includes standard namespace declarations as well as **policy** specific
1501 parameters, such as PolicyId and RuleCombiningAlgId.

1502 [h8] **Policy** identifier. This parameter allows the **policy** to be referenced by a **policy set**.

1503 [h10] The **Rule-combining algorithm** identifies the algorithm for combining the outcomes of **rule**
1504 evaluation.

1505 [h11] - [h14] Free-form description of the **policy**.

1506 [h18] - [h33] **Policy target**. The **policy target** defines a set of applicable **decision requests**. The
1507 structure of the <Target> element in the <Policy> is identical to the structure of the <Target>

1508 element in the <Rule>. In this case, the **policy target** is the set of all XML **resources** that conform to
1509 the namespace “urn:example:med:schemas:record”.

1510 [h34] - [h108] The only <Rule> element included in this <Policy>. Two parameters are specified in the
1511 **rule** header: RuleId and Effect.

1512 [h41] - [h87] The **rule target** further constrains the **policy target**.

1513 [h44] - [h53] The <Match> element targets the **rule** at **subjects** whose
1514 “urn:oasis:names:tc:xacml:3.0:example:attribute:role” **subject attribute** is equal to “physician”.

1515 [h58] - [h69] The <Match> element targets the **rule** at **resources** that match the XPath expression
1516 “md:record/md:medical”.

1517 [h74] - [h84] The <Match> element targets the **rule** at **actions** whose
1518 “urn:oasis:names:tc:xacml:1.0:action:action-id” **action attribute** is equal to “write”.

1519 [h88] - [h107] The <Condition> element. For the **rule** to be applicable to the **decision request**, the
1520 **condition** must evaluate to “True”. This **condition** compares the value of the
1521 “urn:oasis:names:tc:xacml:3.0:example:attribute:physician-id” **subject attribute** with the value of the
1522 <registrationId> element in the medical record that is being accessed.

1523 [h109] - [h134] The <ObligationExpressions> element. **Obligations** are a set of operations that
1524 must be performed by the **PEP** in conjunction with an **authorization decision**. An **obligation** may be
1525 associated with a “Permit” or “Deny” **authorization decision**. The element contains a single **obligation**
1526 expression, which will be evaluated into an obligation when the policy is evaluated.

1527 [h110] - [h133] The <ObligationExpression> element consists of the ObligationId attribute, the
1528 **authorization decision** value for which it must be fulfilled, and a set of **attribute** assignments.

1529 [h110] The ObligationId attribute identifies the **obligation**. In this case, the **PEP** is required to send
1530 email.

1531 [h111] The FulfillOn attribute defines the **authorization decision** value for which the **obligation**
1532 derived from the **obligation** expression must be fulfilled. In this case, the **obligation** must be fulfilled
1533 when **access** is permitted.

1534 [h112] - [h119] The first parameter indicates where the **PEP** will find the email address in the **resource**.
1535 The **PDP** will evaluate the <AttributeSelector> and return the result to the **PEP** inside the resulting
1536 **obligation**.

1537 [h120] - [h123] The second parameter contains literal text for the email body.

1538 [h125] - [h132] The third parameter indicates where the **PEP** will find further text for the email body in the
1539 **resource**. The **PDP** will evaluate the <AttributeDesignator> and return the result to the **PEP** inside
1540 the resulting **obligation**.

1541 4.2.4.4 Rule 4

1542 **Rule 4** illustrates the use of the “Deny” **Effect** value, and a <Rule> with no <Condition> element.

```

1543 [i1] <?xml version="1.0" encoding="UTF-8"?>
1544 [i2] <Policy
1545 [i3]   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1546 [i4]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1547 [i5]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1548 [i6]   PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:4"
1549 [i7]   Version="1.0"
1550 [i8]   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1551 [i9]   algorithm:deny-overrides">
1552 [i9]   <PolicyDefaults>
1553 [i10]     <XPathVersion>http://www.w3.org/TR/1999/REC-xpath-19991116</XPathVersion>
1554 [i11]   </PolicyDefaults>
1555 [i12]   <Target/>
1556 [i13]   <Rule
1557 [i14]     RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:4"
1558 [i15]     Effect="Deny">
1559 [i16]   <Description>
1560 [i17]     An Administrator shall not be permitted to read or write

```

```

1561 [i18] medical elements of a patient record in the
1562 [i19] http://www.med.example.com/records.xsd namespace.
1563 [i20] </Description>
1564 [i21] <Target>
1565 [i22] <AnyOf>
1566 [i23] <AllOf>
1567 [i24] <Match
1568 [i25] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1569 [i26] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1570 [i27] >administrator</AttributeValue>
1571 [i28] <AttributeDesignator
1572 [i29] MustBePresent="false"
1573 [i30] Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
1574 [i31] AttributeId="urn:oasis:names:tc:xacml:3.0:example:attribute:role"
1575 [i32] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1576 [i33] </Match>
1577 [i34] </AllOf>
1578 [i35] </AnyOf>
1579 [i36] <AnyOf>
1580 [i37] <AllOf>
1581 [i38] <Match
1582 [i39] MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
1583 [i40] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1584 [i41] >urn:example:med:schemas:record</AttributeValue>
1585 [i42] <AttributeDesignator
1586 [i43] MustBePresent="false"
1587 [i44] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1588 [i45] AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1589 [i46] DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
1590 [i47] </Match>
1591 [i48] <Match
1592 [i49] MatchId="urn:oasis:names:tc:xacml:3.0:function:xpath-node-match">
1593 [i50] <AttributeValue
1594 [i51] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"
1595 [i52] XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1596 [i53] >md:record/md:medical</AttributeValue>
1597 [i54] <AttributeDesignator
1598 [i55] MustBePresent="false"
1599 [i56] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1600 [i57] AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector"
1601 [i58] DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"/>
1602 [i59] </Match>
1603 [i60] </AllOf>
1604 [i61] </AnyOf>
1605 [i62] <AnyOf>
1606 [i63] <AllOf>
1607 [i64] <Match
1608 [i65] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1609 [i66] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1610 [i67] >read</AttributeValue>
1611 [i68] <AttributeDesignator
1612 [i69] MustBePresent="false"
1613 [i70] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1614 [i71] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1615 [i72] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1616 [i73] </Match>
1617 [i74] </AllOf>
1618 [i75] <AllOf>
1619 [i76] <Match
1620 [i77] MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1621 [i78] <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1622 [i79] >write</AttributeValue>
1623 [i80] <AttributeDesignator
1624 [i81] MustBePresent="false"
1625 [i82] Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
1626 [i83] AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1627 [i84] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1628 [i85] </Match>
1629 [i86] </AllOf>
1630 [i87] </AnyOf>
1631 [i88] </Target>
1632 [i89] </Rule>
1633 [i90] </Policy>

```


1634 [i13] - [i15] The <Rule> element declaration.

1635 [i15] **Rule Effect**. Every **rule** that evaluates to “True” emits the **rule effect** as its value. This **rule**
 1636 **Effect** is “Deny” meaning that according to this **rule**, **access** must be denied when it evaluates to
 1637 “True”.

1638 [i16] - [i20] Free form description of the **rule**.

1639 [i21] - [i88] **Rule target**. The **Rule target** defines the set of **decision requests** that are applicable to the
 1640 **rule**.

1641 [i24] - [i33] The <Match> element targets the **rule** at **subjects** whose
 1642 “urn:oasis:names:tc:xacml:3.0:example:attribute:role” **subject attribute** is equal to “administrator”.

1643 [i36] - [i61] The <AnyOf> element contains one <AllOf> element, which (in turn) contains two <Match>
 1644 elements. The **target** matches if the **resource** identified by the request **context** matches both **resource**
 1645 match criteria.

1646 [i38] - [i47] The first <Match> element targets the **rule** at **resources** whose
 1647 “urn:oasis:names:tc:xacml:2.0:resource:target-namespace” **resource attribute** is equal to
 1648 “urn:example:med:schemas:record”.

1649 [i48] - [i59] The second <Match> element targets the **rule** at XML elements that match the XPath
 1650 expression “/md:record/md:medical”.

1651 [i62] - [i87] The <AnyOf> element contains two <AllOf> elements, each of which contains one <Match>
 1652 element. The **target** matches if the **action** identified in the request **context** matches either of the **action**
 1653 match criteria.

1654 [i64] - [i85] The <Match> elements **target** the **rule** at **actions** whose
 1655 “urn:oasis:names:tc:xacml:1.0:action:action-id” **action attribute** is equal to “read” or “write”.

1656 This **rule** does not have a <Condition> element.

1657 4.2.4.5 Example PolicySet

1658 This section uses the examples of the previous sections to illustrate the process of combining **policies**.
 1659 The **policy** governing read **access** to medical elements of a record is formed from each of the four **rules**
 1660 described in Section 4.2.3. In plain language, the combined **rule** is:

- 1661 • Either the requestor is the patient; or
- 1662 • the requestor is the parent or guardian and the patient is under 16; or
- 1663 • the requestor is the primary care physician and a notification is sent to the patient; and
- 1664 • the requestor is not an administrator.

1665 The following **policy set** illustrates the combined **policies**. **Policy 3** is included by reference and **policy**
 1666 **2** is explicitly included.

```

1667 [j1]    <?xml version="1.0" encoding="UTF-8"?>
1668 [j2]    <PolicySet
1669 [j3]      xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
1670 [j4]      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1671 [j5]      PolicySetId="urn:oasis:names:tc:xacml:3.0:example:policysetid:1"
1672 [j6]      Version="1.0"
1673 [j7]      PolicyCombiningAlgId=
1674 [j8]      "urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides">
1675 [j9]    <Description>
1676 [j10]     Example policy set.
1677 [j11]    </Description>
1678 [j12]    <Target>
1679 [j13]     <AnyOf>
1680 [j14]       <AllOf>
1681 [j15]         <Match
1682 [j16]           MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1683 [j17]           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"
1684 [j18]             >urn:example:med:schema:records</AttributeValue>
1685 [j19]           <AttributeDesignator
1686 [j20]             MustBePresent="false"

```

```

1687 [j21]         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
1688 [j22]         AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1689 [j23]         DataType="http://www.w3.org/2001/XMLSchema#string"/>
1690 [j24]         </Match>
1691 [j25]         </AllOf>
1692 [j26]         </AnyOf>
1693 [j27]         </Target>
1694 [j28]         <PolicyIdReference>
1695 [j29]           urn:oasis:names:tc:xacml:3.0:example:policyid:3
1696 [j30]         </PolicyIdReference>
1697 [j31]         <Policy
1698 [j32]           PolicyId="urn:oasis:names:tc:xacml:3.0:example:policyid:2"
1699 [j33]           RuleCombiningAlgId=
1700 [j34]             "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides"
1701 [j35]           Version="1.0">
1702 [j36]           <Target/>
1703 [j37]           <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:1"
1704 [j38]             Effect="Permit">
1705 [j39]           </Rule>
1706 [j40]           <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:2"
1707 [j41]             Effect="Permit">
1708 [j42]           </Rule>
1709 [j43]           <Rule RuleId="urn:oasis:names:tc:xacml:3.0:example:ruleid:4"
1710 [j44]             Effect="Deny">
1711 [j45]           </Rule>
1712 [j46]         </Policy>
1713 [j47]       </PolicySet>

```

1714 [j2] - [j8] The <PolicySet> element declaration. Standard XML namespace declarations are included.

1715 [j5] The PolicySetId attribute is used for identifying this **policy set** for possible inclusion in another
1716 **policy set**.

1717 [j7] - [j8] The **policy-combining algorithm** identifier. **Policies** and **policy sets** in this **policy set** are
1718 combined according to the specified **policy-combining algorithm** when the **authorization decision** is
1719 computed.

1720 [j9] - [j11] Free form description of the **policy set**.

1721 [j12] - [j27] The **policy set** <Target> element defines the set of **decision requests** that are applicable to
1722 this <PolicySet> element.

1723 [j28] - [j30] PolicyIdReference includes a **policy** by id.

1724 [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>
```

```

1774 <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
1775 <xs:attribute name="Version" type="xacml:VersionType" use="required"/>
1776 <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
1777 <xs:attribute name="MaxDelegationDepth" type="xs:integer" use="optional"/>
1778 </xs:complexType>

```

1779 The <PolicySet> element is of PolicySetType complex type.

1780 The <PolicySet> element contains the following attributes and elements:

1781 PolicySetId [Required]

1782 **Policy set** identifier. It is the responsibility of the **PAP** to ensure that no two **policies** visible to
 1783 the **PDP** have the same identifier. This MAY be achieved by following a predefined URN or URI
 1784 scheme. If the **policy set** identifier is in the form of a URL, then it MAY be resolvable.

1785 Version [Required]

1786 The version number of the PolicySet.

1787 PolicyCombiningAlgId [Required]

1788 The identifier of the **policy-combining algorithm** by which the <PolicySet>,
 1789 <CombinerParameters>, <PolicyCombinerParameters> and
 1790 <PolicySetCombinerParameters> components MUST be combined. Standard **policy-**
 1791 **combining algorithms** are listed in Appendix Appendix C. Standard **policy-combining**
 1792 **algorithm** identifiers are listed in Section B.9.

1793 MaxDelegationDepth [Optional]

1794 If present, limits the depth of delegation which is authorized by this **policy set**. See the delegation
 1795 profile [XACMLAdmin].

1796 <Description> [Optional]

1797 A free-form description of the **policy set**.

1798 <PolicyIssuer> [Optional]

1799 **Attributes** of the **issuer** of the **policy set**.

1800 <PolicySetDefaults> [Optional]

1801 A set of default values applicable to the **policy set**. The scope of the <PolicySetDefaults>
 1802 element SHALL be the enclosing **policy set**.

1803 <Target> [Required]

1804 The <Target> element defines the applicability of a **policy set** to a set of **decision requests**.

1805 The <Target> element MAY be declared by the creator of the <PolicySet> or it MAY be computed
 1806 from the <Target> elements of the referenced <Policy> elements, either as an intersection or
 1807 as a union.

1808 <PolicySet> [Any Number]

1809 A **policy set** that is included in this **policy set**.

1810 <Policy> [Any Number]

1811 A **policy** that is included in this **policy set**.

1812 <PolicySetIdReference> [Any Number]

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

1815 <PolicyIdReference> [Any Number]

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

- 1818 <ObligationExpressions> [Optional]
 1819 Contains the set of <ObligationExpression> elements. See Section 7.18 for a description of
 1820 how the set of **obligations** to be returned by the **PDP** shall be determined.
- 1821 <AdviceExpressions> [Optional]
 1822 Contains the set of <AdviceExpression> elements. See Section 7.18 for a description of how
 1823 the set of **advice** to be returned by the **PDP** shall be determined.
- 1824 <CombinerParameters> [Optional]
 1825 Contains a sequence of <CombinerParameter> elements. The parameters apply to the
 1826 combining algorithm as such and it is up to the specific combining algorithm to interpret them and
 1827 adjust its behavior accordingly.
- 1828 <PolicyCombinerParameters> [Optional]
 1829 Contains a sequence of <CombinerParameter> elements that are associated with a particular
 1830 <Policy> or <PolicyIdReference> element within the <PolicySet>. It is up to the specific
 1831 combining algorithm to interpret them and adjust its behavior accordingly.
- 1832 <PolicySetCombinerParameters> [Optional]
 1833 Contains a sequence of <CombinerParameter> elements that are associated with a particular
 1834 <PolicySet> or <PolicySetIdReference> element within the <PolicySet>. It is up to the
 1835 specific combining algorithm to interpret them and adjust its behavior accordingly.

1836 5.2 Element <Description>

- 1837 The <Description> element contains a free-form description of the <PolicySet>, <Policy>,
 1838 <Rule> or <Apply> element. The <Description> element is of xs:string simple type.

```
1839 <xs:element name="Description" type="xs:string"/>
```

1840 5.3 Element <PolicyIssuer>

- 1841 The <PolicyIssuer> element contains **attributes** describing the issuer of the **policy** or **policy set**.
 1842 The use of the **policy** issuer element is defined in a separate administration profile [**XACMLAdmin**]. A
 1843 PDP which does not implement the administration profile **MUST** report an error or return an Indeterminate
 1844 result if it encounters this element.

```
1845 <xs:element name="PolicyIssuer" type="xacml:PolicyIssuerType"/>
1846 <xs:complexType name="PolicyIssuerType">
1847   <xs:sequence>
1848     <xs:element ref="xacml:Content" minOccurs="0"/>
1849     <xs:element ref="xacml:Attribute" minOccurs="0" maxOccurs="unbounded"/>
1850   </xs:sequence>
1851 </xs:complexType>
```

- 1852 The <PolicyIssuer> element is of PolicyIssuerType complex type.

- 1853 The <PolicyIssuer> element contains the following elements:

1854 <Content> [Optional]

- 1855 Free form XML describing the issuer. See Section 5.45.

1856 <Attribute> [Zero to many]

- 1857 An **attribute** of the issuer. See Section 5.46.

1858 5.4 Element <PolicySetDefaults>

- 1859 The <PolicySetDefaults> element **SHALL** specify default values that apply to the <PolicySet>
 1860 element.

1861
1862
1863
1864
1865
1866
1867
1868

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

1869 <PolicySetDefaults> element is of DefaultsType complex type.
1870 The <PolicySetDefaults> element contains the following elements:
1871 <XPathVersion> [Optional]
1872 Default XPath version.

1873 5.5 Element <XPathVersion>

1874 The <XPathVersion> element SHALL specify the version of the XPath specification to be used by
1875 <AttributeSelector> elements and XPath-based functions in the **policy set** or **policy**.

1876

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

1877 The URI for the XPath 1.0 specification is "http://www.w3.org/TR/1999/REC-xpath-19991116".
1878 The URI for the XPath 2.0 specification is "http://www.w3.org/TR/2007/REC-xpath20-20070123".
1879 The <XPathVersion> element is REQUIRED if the XACML enclosing **policy set** or **policy** contains
1880 <AttributeSelector> elements or XPath-based functions.

1881 5.6 Element <Target>

1882 The <Target> element identifies the set of **decision requests** that the parent element is intended to
1883 evaluate. The <Target> element SHALL appear as a child of a <PolicySet> and <Policy> element
1884 and MAY appear as a child of a <Rule> element.

1885 The <Target> element SHALL contain a **conjunctive sequence** of <AnyOf> elements. For the parent
1886 of the <Target> element to be applicable to the **decision request**, there MUST be at least one positive
1887 match between each <AnyOf> element of the <Target> element and the corresponding section of the
1888 <Request> element.

1889
1890
1891
1892
1893
1894

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

1895 The <Target> element is of TargetType complex type.
1896 The <Target> element contains the following elements:
1897 <AnyOf> [Zero to Many]
1898 Matching specification for **attributes** in the **context**. If this element is missing, then the **target**
1899 SHALL match all **contexts**.

1900 5.7 Element <AnyOf>

1901 The <AnyOf> element SHALL contain a **disjunctive sequence** of <AllOf> elements.

1902
1903
1904
1905

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

1906 `</xs:sequence>`
1907 `</xs:complexType>`

1908 The `<AnyOf>` element is of `AnyOfType` complex type.

1909 The `<AnyOf>` element contains the following elements:

1910 `<AllOf>` [One to Many, Required]

1911 See Section 5.8.

1912 5.8 Element `<AllOf>`

1913 The `<AllOf>` element SHALL contain a **conjunctive sequence** of `<Match>` elements.

```
1914 <xs:element name="AllOf" type="xacml:AllOfType"/>
1915 <xs:complexType name="AllOfType">
1916   <xs:sequence minOccurs="1" maxOccurs="unbounded">
1917     <xs:element ref="xacml:Match"/>
1918   </xs:sequence>
1919 </xs:complexType>
```

1920 The `<AllOf>` element is of `AllOfType` complex type.

1921 The `<AllOf>` element contains the following elements:

1922 `<Match>` [One to Many]

1923 A **conjunctive sequence** of individual matches of the **attributes** in the request **context** and the
1924 embedded **attribute** values. See Section 5.9.

1925 5.9 Element `<Match>`

1926 The `<Match>` element SHALL identify a set of entities by matching **attribute** values in an

1927 `<Attributes>` element of the request **context** with the embedded **attribute** value.

```
1928 <xs:element name="Match" type="xacml:MatchType"/>
1929 <xs:complexType name="MatchType">
1930   <xs:sequence>
1931     <xs:element ref="xacml:AttributeValue"/>
1932     <xs:choice>
1933       <xs:element ref="xacml:AttributeDesignator"/>
1934       <xs:element ref="xacml:AttributeSelector"/>
1935     </xs:choice>
1936   </xs:sequence>
1937   <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
1938 </xs:complexType>
```

1939 The `<Match>` element is of `MatchType` complex type.

1940 The `<Match>` element contains the following attributes and elements:

1941 `MatchId` [Required]

1942 Specifies a matching function. The value of this attribute MUST be of type `xs:anyURI` with legal
1943 values documented in Section 7.6.

1944 `<AttributeValue>` [Required]

1945 Embedded **attribute** value.

1946 `<AttributeDesignator>` [Required choice]

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

1949 `<AttributeSelector>` [Required choice]

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

1952 5.10 Element <PolicySetIdReference>

1953 The <PolicySetIdReference> element SHALL be used to reference a <PolicySet> element by id.
1954 If <PolicySetIdReference> is a URL, then it MAY be resolvable to the <PolicySet> element.
1955 However, the mechanism for resolving a **policy set** reference to the corresponding **policy set** is outside
1956 the scope of this specification.

```
1957 <xs:element name="PolicySetIdReference" type="xacml:IdReferenceType"/>  
1958 <xs:complexType name="IdReferenceType">  
1959   <xs:simpleContent>  
1960     <xs:extension base="xs:anyURI">  
1961       <xs:attribute name="xacml:Version"  
1962         type="xacml:VersionMatchType" use="optional"/>  
1963       <xs:attribute name="xacml:EarliestVersion"  
1964         type="xacml:VersionMatchType" use="optional"/>  
1965       <xs:attribute name="xacml:LatestVersion"  
1966         type="xacml:VersionMatchType" use="optional"/>  
1967     </xs:extension>  
1968   </xs:simpleContent>  
1969 </xs:complexType>
```

1970 Element <PolicySetIdReference> is of xacml:IdReferenceType complex type.

1971 IdReferenceType extends the xs:anyURI type with the following attributes:

1972 Version [Optional]

1973 Specifies a matching expression for the version of the **policy set** referenced.

1974 EarliestVersion [Optional]

1975 Specifies a matching expression for the earliest acceptable version of the **policy set** referenced.

1976 LatestVersion [Optional]

1977 Specifies a matching expression for the latest acceptable version of the **policy set** referenced.

1978 The matching operation is defined in Section 5.13. Any combination of these attributes MAY be present
1979 in a <PolicySetIdReference>. The referenced **policy set** MUST match all expressions. If none of
1980 these attributes is present, then any version of the **policy set** is acceptable. In the case that more than
1981 one matching version can be obtained, then the most recent one SHOULD be used.

1982 5.11 Element <PolicyIdReference>

1983 The <PolicyIdReference> element SHALL be used to reference a <Policy> element by id. If
1984 <PolicyIdReference> is a URL, then it MAY be resolvable to the <Policy> element. However, the
1985 mechanism for resolving a **policy** reference to the corresponding **policy** is outside the scope of this
1986 specification.

```
1987 <xs:element name="PolicyIdReference" type="xacml:IdReferenceType"/>
```

1988 Element <PolicyIdReference> is of xacml:IdReferenceType complex type (see Section 5.10) .

1989 5.12 Simple type VersionType

1990 Elements of this type SHALL contain the version number of the **policy** or **policy set**.

```
1991 <xs:simpleType name="VersionType">  
1992   <xs:restriction base="xs:string">  
1993     <xs:pattern value="(\d+\.)*\d+"/>  
1994   </xs:restriction>  
1995 </xs:simpleType>
```


1996 The version number is expressed as a sequence of decimal numbers, each separated by a period (.).
1997 'd+' represents a sequence of one or more decimal digits.

1998 5.13 Simple type VersionMatchType

1999 Elements of this type SHALL contain a restricted regular expression matching a version number (see
2000 Section 5.12). The expression SHALL match versions of a referenced **policy** or **policy set** that are
2001 acceptable for inclusion in the referencing **policy** or **policy set**.

```
2002 <xs:simpleType name="VersionMatchType">  
2003   <xs:restriction base="xs:string">  
2004     <xs:pattern value="((\d+|\*)\.)*(\d+|\*|\+)" />  
2005   </xs:restriction>  
2006 </xs:simpleType>
```

2007 A version match is '.'-separated, like a version string. A number represents a direct numeric match. A '*'
2008 means that any single number is valid. A '+' means that any number, and any subsequent numbers, are
2009 valid. In this manner, the following four patterns would all match the version string '1.2.3': '1.2.3', '1.*.3',
2010 '1.2.*' and '1.+'

2011 5.14 Element <Policy>

2012 The <Policy> element is the smallest entity that SHALL be presented to the **PDP** for evaluation.

2013 A <Policy> element may be evaluated, in which case the evaluation procedure defined in Section 7.12
2014 SHALL be used.

2015 The main components of this element are the <Target>, <Rule>, <CombinerParameters>,
2016 <RuleCombinerParameters>, <ObligationExpressions> and <AdviceExpressions>
2017 elements and the RuleCombiningAlgId attribute.

2018 A <Policy> element MAY contain a <PolicyIssuer> element. The interpretation of the
2019 <PolicyIssuer> element is explained in the separate administrative **policy** profile [XACMLAdmin].

2020 The <Target> element defines the applicability of the <Policy> element to a set of **decision requests**.
2021 If the <Target> element within the <Policy> element matches the request **context**, then the
2022 <Policy> element MAY be used by the **PDP** in making its **authorization decision**. See Section 7.12.

2023 The <Policy> element includes a sequence of choices between <VariableDefinition> and
2024 <Rule> elements.

2025 **Rules** included in the <Policy> element MUST be combined by the algorithm specified by the
2026 RuleCombiningAlgId attribute.

2027 The <ObligationExpressions> element contains a set of **obligation** expressions that MUST be
2028 evaluated into **obligations** by the **PDP** and the resulting **obligations** MUST be fulfilled by the **PEP** in
2029 conjunction with the **authorization decision**. If the **PEP** does not understand, or cannot fulfill, any of the
2030 **obligations**, then it MUST act according to the PEP bias. See Section 7.2 and 7.18.

2031 The <AdviceExpressions> element contains a set of **advice** expressions that MUST be evaluated into
2032 **advice** by the **PDP**. The resulting **advice** MAY be safely ignored by the **PEP** in conjunction with the
2033 **authorization decision**. See Section 7.18.

```
2034 <xs:element name="Policy" type="xacml:PolicyType"/>  
2035 <xs:complexType name="PolicyType">  
2036   <xs:sequence>  
2037     <xs:element ref="xacml:Description" minOccurs="0"/>  
2038     <xs:element ref="xacml:PolicyIssuer" minOccurs="0"/>  
2039     <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>  
2040     <xs:element ref="xacml:Target"/>  
2041     <xs:choice maxOccurs="unbounded">  
2042       <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>  
2043       <xs:element ref="xacml:RuleCombinerParameters" minOccurs="0"/>  
2044       <xs:element ref="xacml:VariableDefinition"/>
```

```

2045         <xs:element ref="xacml:Rule"/>
2046     </xs:choice>
2047     <xs:element ref="xacml:ObligationExpressions" minOccurs="0"/>
2048     <xs:element ref="xacml:AdviceExpressions" minOccurs="0"/>
2049 </xs:sequence>
2050 <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
2051 <xs:attribute name="Version" type="xacml:VersionType" use="required"/>
2052 <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
2053 <xs:attribute name="MaxDelegationDepth" type="xs:integer" use="optional"/>
2054 </xs:complexType>

```

2055 The <Policy> element is of PolicyType complex type.

2056 The <Policy> element contains the following attributes and elements:

2057 PolicyId [Required]

2058 **Policy** identifier. It is the responsibility of the **PAP** to ensure that no two **policies** visible to the
2059 **PDP** have the same identifier. This MAY be achieved by following a predefined URN or URI
2060 scheme. If the **policy** identifier is in the form of a URL, then it MAY be resolvable.

2061 Version [Required]

2062 The version number of the **Policy**.

2063 RuleCombiningAlgId [Required]

2064 The identifier of the **rule-combining algorithm** by which the <Policy>,
2065 <CombinerParameters> and <RuleCombinerParameters> components MUST be
2066 combined. Standard **rule-combining algorithms** are listed in Appendix Appendix C. Standard
2067 **rule-combining algorithm** identifiers are listed in Section B.9.

2068 MaxDelegationDepth [Optional]

2069 If present, limits the depth of delegation which is authorized by this **policy**. See the delegation
2070 profile [**XACMLAdmin**].

2071 <Description> [Optional]

2072 A free-form description of the **policy**. See Section 5.2.

2073 <PolicyIssuer> [Optional]

2074 **Attributes** of the **issuer** of the **policy**.

2075 <PolicyDefaults> [Optional]

2076 Defines a set of default values applicable to the **policy**. The scope of the <PolicyDefaults>
2077 element SHALL be the enclosing **policy**.

2078 <CombinerParameters> [Optional]

2079 A sequence of parameters to be used by the **rule-combining algorithm**. The parameters apply
2080 to the combining algorithm as such and it is up to the specific combining algorithm to interpret
2081 them and adjust its behavior accordingly.

2082 <RuleCombinerParameters> [Optional]

2083 A sequence of <RuleCombinerParameter> elements that are associated with a particular
2084 <Rule> element within the <Policy>.. It is up to the specific combining algorithm to interpret
2085 them and adjust its behavior accordingly.

2086 <Target> [Required]

2087 The <Target> element defines the applicability of a <Policy> to a set of **decision requests**.

2088 The <Target> element MAY be declared by the creator of the <Policy> element, or it MAY be
2089 computed from the <Target> elements of the referenced <Rule> elements either as an
2090 intersection or as a union.

2091 <VariableDefinition> [Any Number]
 2092 Common function definitions that can be referenced from anywhere in a *rule* where an
 2093 expression can be found.

2094 <Rule> [Any Number]
 2095 A sequence of *rules* that MUST be combined according to the RuleCombiningAlgId attribute.
 2096 *Rules* whose <Target> elements and conditions match the *decision request* MUST be
 2097 considered. *Rules* whose <Target> elements or conditions do not match the *decision request*
 2098 SHALL be ignored.

2099 <ObligationExpressions> [Optional]
 2100 A *conjunctive sequence* of *obligation* expressions which MUST be evaluated into *obligations*
 2101 ~~by~~ by the PDP. The ~~corresponding~~ *corresponding obligations* MUST be fulfilled by the *PEP* in
 2102 conjunction with the *authorization decision*. See Section 7.18 for a description of how the set of
 2103 *obligations* to be returned by the *PDP* SHALL be determined. See section 7.2 about
 2104 enforcement of *obligations*.

2105 <AdviceExpressions> [Optional]
 2106 A *conjunctive sequence* of *advice* expressions which MUST evaluated into *advice* by the *PDP*.
 2107 The corresponding *advice* provide supplementary information to the *PEP* in conjunction with the
 2108 *authorization decision*. See Section 7.18 for a description of how the set of *advice* to be
 2109 returned by the *PDP* SHALL be determined.

2110 5.15 Element <PolicyDefaults>

2111 The <PolicyDefaults> element SHALL specify default values that apply to the <Policy> element.

```

2112 <xs:element name="PolicyDefaults" type="xacml:DefaultsType"/>
2113 <xs:complexType name="DefaultsType">
2114   <xs:sequence>
2115     <xs:choice>
2116       <xs:element ref="xacml:XPathVersion" />
2117     </xs:choice>
2118   </xs:sequence>
2119 </xs:complexType>
  
```

2120 <PolicyDefaults> element is of DefaultsType complex type.

2121 The <PolicyDefaults> element contains the following elements:

2122 <XPathVersion> [Optional]

2123 Default XPath version.

2124 5.16 Element <CombinerParameters>

2125 The <CombinerParameters> element conveys parameters for a *policy- or rule-combining algorithm*.

2126 If multiple <CombinerParameters> elements occur within the same *policy* or *policy set*, they SHALL
 2127 be considered equal to one <CombinerParameters> element containing the concatenation of all the
 2128 sequences of <CombinerParameters> contained in all the aforementioned <CombinerParameters>
 2129 elements, such that the order of ~~occurrence~~ *occurrence* of the
 2130 ~~<CombinerParameters>~~ *CombinerParameters* elements is preserved in the concatenation of the
 2131 <CombinerParameter> elements.

2132 Note that none of the combining algorithms specified in XACML 3.0 is parameterized.

```

2133 <xs:element name="CombinerParameters" type="xacml:CombinerParametersType"/>
2134 <xs:complexType name="CombinerParametersType">
2135   <xs:sequence>
2136     <xs:element ref="xacml:CombinerParameter" minOccurs="0"
  
```

```
2137         maxOccurs="unbounded"/>
2138     </xs:sequence>
2139 </xs:complexType>
```

2140 The <CombinerParameters> element is of CombinerParametersType complex type.

2141 The <CombinerParameters> element contains the following elements:

2142 <CombinerParameter> [Any Number]

2143 A single parameter. See Section 5.17.

2144 Support for the <CombinerParameters> element is optional.

2145 5.17 Element <CombinerParameter>

2146 The <CombinerParameter> element conveys a single parameter for a *policy*- or *rule-combining*
2147 *algorithm*.

```
2148 <xs:element name="CombinerParameter" type="xacml:CombinerParameterType"/>
2149 <xs:complexType name="CombinerParameterType">
2150     <xs:sequence>
2151         <xs:element ref="xacml:AttributeValue"/>
2152     </xs:sequence>
2153     <xs:attribute name="ParameterName" type="xs:string" use="required"/>
2154 </xs:complexType>
```

2155 The <CombinerParameter> element is of CombinerParameterType complex type.

2156 The <CombinerParameter> element contains the following attributes:

2157 ParameterName [Required]

2158 The identifier of the parameter.

2159 <AttributeValue> [Required]

2160 The value of the parameter.

2161 Support for the <CombinerParameter> element is optional.

2162 5.18 Element <RuleCombinerParameters>

2163 The <RuleCombinerParameters> element conveys parameters associated with a particular *rule*
2164 within a *policy* for a *rule-combining algorithm*.

2165 Each <RuleCombinerParameters> element MUST be associated with a *rule* contained within the
2166 same *policy*. If multiple <RuleCombinerParameters> elements reference the same *rule*, they SHALL
2167 be considered equal to one <RuleCombinerParameters> element containing the concatenation of all
2168 the sequences of <CombinerParameters> contained in all the aforementioned
2169 <RuleCombinerParameters> elements, such that the order of occurrence of the
2170 ~~<RuleCombinerParameters>~~RuleCombinerParameters elements is preserved in the
2171 concatenation of the <CombinerParameter> elements.

2172 Note that none of the *rule-combining algorithms* specified in XACML 3.0 is parameterized.

```
2173 <xs:element name="RuleCombinerParameters"
2174 type="xacml:RuleCombinerParametersType"/>
2175 <xs:complexType name="RuleCombinerParametersType">
2176     <xs:complexContent>
2177         <xs:extension base="xacml:CombinerParametersType">
2178             <xs:attribute name="RuleIdRef" type="xs:string"
2179                 use="required"/>
2180         </xs:extension>
2181     </xs:complexContent>
2182 </xs:complexType>
```

2183 The <RuleCombinerParameters> element contains the following attribute:

2184 RuleIdRef [Required]

2185 The identifier of the <Rule> contained in the **policy**.

2186 Support for the <RuleCombinerParameters> element is optional, only if support for combiner
2187 parameters is not implemented.

2188 5.19 Element <PolicyCombinerParameters>

2189 The <PolicyCombinerParameters> element conveys parameters associated with a particular **policy**
2190 within a **policy set** for a **policy-combining algorithm**.

2191 Each <PolicyCombinerParameters> element MUST be associated with a **policy** contained within the
2192 same **policy set**. If multiple <PolicyCombinerParameters> elements reference the same **policy**,
2193 they SHALL be considered equal to one <PolicyCombinerParameters> element containing the
2194 concatenation of all the sequences of <CombinerParameters> contained in all the aforementioned
2195 <PolicyCombinerParameters> elements, such that the order of occurrence of the
2196 ~~<PolicyCombinerParameters>~~PolicyCombinerParameters elements is preserved in the
2197 concatenation of the <CombinerParameter> elements.

2198 Note that none of the **policy-combining algorithms** specified in XACML 3.0 is parameterized.

```
2199 <xs:element name="PolicyCombinerParameters"  
2200 type="xacml:PolicyCombinerParametersType"/>  
2201 <xs:complexType name="PolicyCombinerParametersType">  
2202   <xs:complexContent>  
2203     <xs:extension base="xacml:CombinerParametersType">  
2204       <xs:attribute name="PolicyIdRef" type="xs:anyURI"  
2205 use="required"/>  
2206     </xs:extension>  
2207   </xs:complexContent>  
2208 </xs:complexType>
```

2209 The <PolicyCombinerParameters> element is of PolicyCombinerParametersType complex
2210 type.

2211 The <PolicyCombinerParameters> element contains the following attribute:

2212 PolicyIdRef [Required]

2213 The identifier of a <Policy> or the value of a <PolicyIdReference> contained in the **policy**
2214 **set**.

2215 Support for the <PolicyCombinerParameters> element is optional, only if support for combiner
2216 parameters is not implemented.

2217 5.20 Element <PolicySetCombinerParameters>

2218 The <PolicySetCombinerParameters> element conveys parameters associated with a particular
2219 **policy set** within a **policy set** for a **policy-combining algorithm**.

2220 Each <PolicySetCombinerParameters> element MUST be associated with a **policy set** contained
2221 within the same **policy set**. If multiple <PolicySetCombinerParameters> elements reference the
2222 same **policy set**, they SHALL be considered equal to one <PolicySetCombinerParameters>
2223 element containing the concatenation of all the sequences of <CombinerParameters> contained in all
2224 the aforementioned <PolicySetCombinerParameters> elements, such that the order of occurrence
2225 of the ~~<PolicySetCombinerParameters>~~PolicySetCombinerParameters elements is preserved
2226 in the concatenation of the <CombinerParameter> elements.

2227 Note that none of the **policy-combining algorithms** specified in XACML 3.0 is parameterized.

```

2228 <xs:element name="PolicySetCombinerParameters"
2229 type="xacml:PolicySetCombinerParametersType"/>
2230 <xs:complexType name="PolicySetCombinerParametersType">
2231 <xs:complexContent>
2232 <xs:extension base="xacml:CombinerParametersType">
2233 <xs:attribute name="PolicySetIdRef" type="xs:anyURI"
2234 use="required"/>
2235 </xs:extension>
2236 </xs:complexContent>
2237 </xs:complexType>

```

2238 The <PolicySetCombinerParameters> element is of PolicySetCombinerParametersType
2239 complex type.

2240 The <PolicySetCombinerParameters> element contains the following attribute:

2241 PolicySetIdRef [Required]

2242 The identifier of a <PolicySet> or the value of a <PolicySetIdReference> contained in the
2243 **policy set**.

2244 Support for the <PolicySetCombinerParameters> element is optional, only if support for combiner
2245 parameters is not implemented.

2246 5.21 Element <Rule>

2247 The <Rule> element SHALL define the individual **rules** in the **policy**. The main components of this
2248 element are the <Target>, <Condition>, <ObligationExpressions> and
2249 <AdviceExpressions> elements and the Effect attribute.

2250 A <Rule> element may be evaluated, in which case the evaluation procedure defined in Section 7.10
2251 SHALL be used.

```

2252 <xs:element name="Rule" type="xacml:RuleType"/>
2253 <xs:complexType name="RuleType">
2254 <xs:sequence>
2255 <xs:element ref="xacml:Description" minOccurs="0"/>
2256 <xs:element ref="xacml:Target" minOccurs="0"/>
2257 <xs:element ref="xacml:Condition" minOccurs="0"/>
2258 <xs:element ref="xacml:ObligationExpressions" minOccurs="0"/>
2259 <xs:element ref="xacml:AdviceExpressions" minOccurs="0"/>
2260 </xs:sequence>
2261 <xs:attribute name="RuleId" type="xs:string" use="required"/>
2262 <xs:attribute name="Effect" type="xacml:EffectType" use="required"/>
2263 </xs:complexType>

```

2264 The <Rule> element is of RuleType complex type.

2265 The <Rule> element contains the following attributes and elements:

2266 RuleId [Required]

2267 A string identifying this **rule**.

2268 Effect [Required]

2269 **Rule effect**. The value of this attribute is either "Permit" or "Deny".

2270 <Description> [Optional]

2271 A free-form description of the **rule**.

2272 <Target> [Optional]

2273 Identifies the set of **decision requests** that the <Rule> element is intended to evaluate. If this
2274 element is omitted, then the **target** for the <Rule> SHALL be defined by the <Target> element
2275 of the enclosing <Policy> element. See Section 7.7 for details.

- 2276 <Condition> [Optional]
- 2277 A **predicate** that MUST be satisfied for the **rule** to be assigned its **Effect** value.
- 2278 <ObligationExpressions> [Optional]
- 2279 A **conjunctive sequence** of **obligation** expressions which MUST be evaluated into **obligations**
 2280 ~~by~~by the PDP. The ~~corresponding~~**corresponding** **obligations** MUST be fulfilled by the **PEP** in
 2281 conjunction with the **authorization decision**. See Section 7.18 for a description of how the set of
 2282 **obligations** to be returned by the **PDP** SHALL be determined. See section 7.2 about
 2283 enforcement of **obligations**.
- 2284 <AdviceExpressions> [Optional]
- 2285 A **conjunctive sequence** of **advice** expressions which MUST evaluated into **advice** by the **PDP**.
 2286 The corresponding **advice** provide supplementary information to the **PEP** in conjunction with the
 2287 **authorization decision**. See Section 7.18 for a description of how the set of **advice** to be
 2288 returned by the **PDP** SHALL be determined.

2289 5.22 Simple type EffectType

- 2290 The **EffectType** simple type defines the values allowed for the **Effect** attribute of the <Rule> element
 2291 and for the **FulfillOn** attribute of the <ObligationExpression> and <AdviceExpression>
 2292 elements.

```

2293 <xs:simpleType name="EffectType">
2294   <xs:restriction base="xs:string">
2295     <xs:enumeration value="Permit"/>
2296     <xs:enumeration value="Deny"/>
2297   </xs:restriction>
2298 </xs:simpleType>
  
```

2299 5.23 Element <VariableDefinition>

- 2300 The <VariableDefinition> element SHALL be used to define a value that can be referenced by a
 2301 <VariableReference> element. The name supplied for its **VariableId** attribute SHALL NOT occur
 2302 in the **VariableId** attribute of any other <VariableDefinition> element within the encompassing
 2303 **policy**. The <VariableDefinition> element MAY contain undefined <VariableReference>
 2304 elements, but if it does, a corresponding <VariableDefinition> element MUST be defined later in
 2305 the encompassing **policy**. <VariableDefinition> elements MAY be grouped together or MAY be
 2306 placed close to the reference in the encompassing **policy**. There MAY be zero or more references to
 2307 each <VariableDefinition> element.

```

2308 <xs:element name="VariableDefinition" type="xacml:VariableDefinitionType"/>
2309 <xs:complexType name="VariableDefinitionType">
2310   <xs:sequence>
2311     <xs:element ref="xacml:Expression"/>
2312   </xs:sequence>
2313   <xs:attribute name="VariableId" type="xs:string" use="required"/>
2314 </xs:complexType>
  
```

- 2315 The <VariableDefinition> element is of **VariableDefinitionType** complex type. The
 2316 <VariableDefinition> element has the following elements and attributes:

- 2317 <Expression> [Required]
- 2318 Any element of **ExpressionType** complex type.
- 2319 **VariableId** [Required]
- 2320 The name of the variable definition.

2321 5.24 Element <VariableReference>

2322 The <VariableReference> element is used to reference a value defined within the same
2323 encompassing <Policy> element. The <VariableReference> element SHALL refer to the
2324 <VariableDefinition> element by **identifier equality** on the value of their respective VariableId
2325 attributes. One and only one <VariableDefinition> MUST exist within the same encompassing
2326 <Policy> element to which the <VariableReference> refers. There MAY be zero or more
2327 <VariableReference> elements that refer to the same <VariableDefinition> element.

```
2328 <xs:element name="VariableReference" type="xacml:VariableReferenceType"  
2329 substitutionGroup="xacml:Expression"/>  
2330 <xs:complexType name="VariableReferenceType">  
2331   <xs:complexContent>  
2332     <xs:extension base="xacml:ExpressionType">  
2333       <xs:attribute name="VariableId" type="xs:string"  
2334         use="required"/>  
2335     </xs:extension>  
2336   </xs:complexContent>  
2337 </xs:complexType>
```

2338 The <VariableReference> element is of the VariableReferenceType complex type, which is of
2339 the ExpressionType complex type and is a member of the <Expression> element substitution group.
2340 The <VariableReference> element MAY appear any place where an <Expression> element occurs
2341 in the schema.

2342 The <VariableReference> element has the following attribute:

2343 VariableId [Required]

2344 The name used to refer to the value defined in a <VariableDefinition> element.

2345 5.25 Element <Expression>

2346 The <Expression> element is not used directly in a **policy**. The <Expression> element signifies that
2347 an element that extends the ExpressionType and is a member of the <Expression> element
2348 substitution group SHALL appear in its place.

```
2349 <xs:element name="Expression" type="xacml:ExpressionType" abstract="true"/>  
2350 <xs:complexType name="ExpressionType" abstract="true"/>
```

2351 The following elements are in the <Expression> element substitution group:

2352 <Apply>, <AttributeSelector>, <AttributeValue>, <Function>, <VariableReference> and
2353 <AttributeDesignator>.

2354 5.26 Element <Condition>

2355 The <Condition> element is a Boolean function over **attributes** or functions of **attributes**.

```
2356 <xs:element name="Condition" type="xacml:ConditionType"/>  
2357 <xs:complexType name="ConditionType">  
2358   <xs:sequence>  
2359     <xs:element ref="xacml:Expression"/>  
2360   </xs:sequence>  
2361 </xs:complexType>
```

2362 The <Condition> contains one <Expression> element, with the restriction that the <Expression>
2363 return data-type MUST be "http://www.w3.org/2001/XMLSchema#boolean". Evaluation of the
2364 <Condition> element is described in Section 7.9.

2365 5.27 Element <Apply>

2366 The <Apply> element denotes application of a function to its arguments, thus encoding a function call.
2367 The <Apply> element can be applied to any combination of the members of the <Expression>
2368 element substitution group. See Section 5.25.

```
2369 <xs:element name="Apply" type="xacml:ApplyType"  
2370 substitutionGroup="xacml:Expression"/>  
2371 <xs:complexType name="ApplyType">  
2372   <xs:complexContent>  
2373     <xs:extension base="xacml:ExpressionType">  
2374       <xs:sequence>  
2375         <xs:element ref="xacml:Description" minOccurs="0"/>  
2376         <xs:element ref="xacml:Expression" minOccurs="0"  
2377           maxOccurs="unbounded"/>  
2378       </xs:sequence>  
2379       <xs:attribute name="FunctionId" type="xs:anyURI"  
2380         use="required"/>  
2381     </xs:extension>  
2382   </xs:complexContent>  
2383 </xs:complexType>
```

2384 The <Apply> element is of ApplyType complex type.

2385 The <Apply> element contains the following attributes and elements:

2386 FunctionId [Required]

2387 The identifier of the function to be applied to the arguments. XACML-defined functions are
2388 described in Appendix A.3.

2389 <Description> [Optional]

2390 A free-form description of the <Apply> element.

2391 <Expression> [Optional]

2392 Arguments to the function, which may include other functions.

2393 5.28 Element <Function>

2394 The <Function> element SHALL be used to name a function as an argument to the function defined by
2395 the parent <Apply> element.

```
2396 <xs:element name="Function" type="xacml:FunctionType"  
2397 substitutionGroup="xacml:Expression"/>  
2398 <xs:complexType name="FunctionType">  
2399   <xs:complexContent>  
2400     <xs:extension base="xacml:ExpressionType">  
2401       <xs:attribute name="FunctionId" type="xs:anyURI"  
2402         use="required"/>  
2403     </xs:extension>  
2404   </xs:complexContent>  
2405 </xs:complexType>
```

2406 The <Function> element is of FunctionType complex type.

2407 The <Function> element contains the following attribute:

2408 FunctionId [Required]

2409 The identifier of the function.

2410 5.29 Element <AttributeDesignator>

2411 The <AttributeDesignator> element retrieves a **bag** of values for a **named attribute** from the
2412 request **context**. A **named attribute** SHALL be considered present if there is at least one **attribute** that
2413 matches the criteria set out below.

2414 The <AttributeDesignator> element SHALL return a **bag** containing all the **attribute** values that are
2415 matched by the **named attribute**. In the event that no matching **attribute** is present in the **context**, the
2416 **MustBePresent** attribute governs whether this element returns an empty **bag** or "Indeterminate". See
2417 Section 7.3.5.

2418 The <AttributeDesignator> MAY appear in the <Match> element and MAY be passed to the
2419 <Apply> element as an argument.

2420 The <AttributeDesignator> element is of the AttributeDesignatorType complex type.

```
2421 <xs:element name="AttributeDesignator" type="xacml:AttributeDesignatorType"  
2422 substitutionGroup="xacml:Expression"/>  
2423 <xs:complexType name="AttributeDesignatorType">  
2424   <xs:complexContent>  
2425     <xs:extension base="xacml:ExpressionType">  
2426       <xs:attribute name="Category" type="xs:anyURI"  
2427         use="required"/>  
2428       <xs:attribute name="AttributeId" type="xs:anyURI"  
2429         use="required"/>  
2430       <xs:attribute name="DataType" type="xs:anyURI"  
2431         use="required"/>  
2432       <xs:attribute name="Issuer" type="xs:string" use="optional"/>  
2433       <xs:attribute name="MustBePresent" type="xs:boolean"  
2434         use="required"/>  
2435     </xs:extension>  
2436   </xs:complexContent>  
2437 </xs:complexType>
```

2438 A **named attribute** SHALL match an **attribute** if the values of their respective **Category**,
2439 **AttributeId**, **DataType** and **Issuer** attributes match. The attribute designator's **Category** MUST
2440 match, by **identifier equality**, the **Category** of the <Attributes> element in which the **attribute** is
2441 present. The attribute designator's **AttributeId** MUST match, by **identifier equality**, the
2442 **AttributeId** of the attribute. The attribute designator's **DataType** MUST match, by **identifier**
2443 **equality**, the **DataType** of the same **attribute**.

2444 If the **Issuer** attribute is present in the attribute designator, then it MUST match, using the
2445 "urn:oasis:names:tc:xacml:1.0:function:string-equal" function, the **Issuer** of the same **attribute**. If the
2446 **Issuer** is not present in the attribute designator, then the matching of the **attribute** to the **named**
2447 **attribute** SHALL be governed by **AttributeId** and **DataType** attributes alone.

2448 The <AttributeDesignatorType> contains the following attributes:

2449 **Category** [Required]

2450 This attribute SHALL specify the **Category** with which to match the **attribute**.

2451 **AttributeId** [Required]

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

2453 **DataType** [Required]

2454 The **bag** returned by the <AttributeDesignator> element SHALL contain values of this data-
2455 type.

2456 **Issuer** [Optional]

2457 This attribute, if supplied, SHALL specify the **Issuer** with which to match the **attribute**.

2458 **MustBePresent** [Required]

2459 This attribute governs whether the element returns “Indeterminate” or an empty **bag** in the event
2460 the **named attribute** is absent from the request **context**. See Section 7.3.5. Also see Sections
2461 7.19.2 and 7.19.3.

2462 5.30 Element <AttributeSelector>

2463 The <AttributeSelector> element produces a **bag** of unnamed and uncategorized **attribute** values.
2464 The values shall be constructed from the node(s) selected by applying the XPath expression given by the
2465 element's Path attribute to the XML content indicated by the element's Category attribute. Support for
2466 the <AttributeSelector> element is OPTIONAL.

2467 See section 7.3.7 for details of <AttributeSelector> evaluation.

```
2468 <xs:element name="AttributeSelector" type="xacml:AttributeSelectorType"  
2469 substitutionGroup="xacml:Expression"/>  
2470 <xs:complexType name="AttributeSelectorType">  
2471 <xs:complexContent>  
2472 <xs:extension base="xacml:ExpressionType">  
2473 <xs:attribute name="Category" type="xs:anyURI"  
2474 use="required"/>  
2475 <xs:attribute name="ContextSelectorId" type="xs:anyURI"  
2476 use="optional"/>  
2477 <xs:attribute name="Path" type="xs:string"  
2478 use="required"/>  
2479 <xs:attribute name="DataType" type="xs:anyURI"  
2480 use="required"/>  
2481 <xs:attribute name="MustBePresent" type="xs:boolean"  
2482 use="required"/>  
2483 </xs:extension>  
2484 </xs:complexContent>  
2485 </xs:complexType>
```

2486 The <AttributeSelector> element is of AttributeSelectorType complex type.

2487 The <AttributeSelector> element has the following attributes:

2488 Category [Required]

2489 This attribute SHALL specify the **attributes** category of the <Content> element containing the
2490 XML from which nodes will be selected. It also indicates the **attributes** category containing the
2491 applicable ContextSelectorId attribute, if the element includes a ContextSelectorId xml
2492 attribute.

2493 ContextSelectorId [Optional]

2494 This attribute refers to the **attribute** (by its AttributeId) in the request **context** in the category
2495 given by the Category attribute. The referenced **attribute** MUST have data type
2496 urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression, and must select a single node in the
2497 <Content> element. The XPathCategory attribute of the referenced **attribute** MUST be equal
2498 to the Category attribute of the **attribute selector**.

2499 Path [Required]

2500 This attribute SHALL contain an XPath expression to be evaluated against the specified XML
2501 content. See Section 7.3.7 for details of the XPath evaluation during <AttributeSelector>
2502 processing. The namespace context for the value of the Path attribute is given by the [in-scope
2503 namespaces] [INFOSET] of the <AttributeSelector> element.

2504 DataType [Required]

2505 The attribute specifies the datatype of the values returned from the evaluation of this
2506 <AttributeSelector> element.

2507 MustBePresent [Required]

2508 This attribute governs whether the element returns “Indeterminate” or an empty **bag** in the event
2509 ~~the XPath expression selects no node.~~ that the attributes category specified by the Category
2510 attribute does not exist in the request context, or the attributes category does exist but it does not
2511 have a <Content> child element, or the <Content> element does exist but the XPath
2512 expression selects no node. See Section 7.3.5. Also see Sections 7.19.2 and 7.19.3.

2513 5.31 Element <AttributeValue>

2514 The <AttributeValue> element SHALL contain a literal **attribute** value.

```
2515 <xs:element name="AttributeValue" type="xacml:AttributeValueType"  
2516 substitutionGroup="xacml:Expression"/>  
2517 <xs:complexType name="AttributeValueType" mixed="true">  
2518   <xs:complexContent mixed="true">  
2519     <xs:extension base="xacml:ExpressionType">  
2520       <xs:sequence>  
2521         <xs:any namespace="##any" processContents="lax"  
2522           minOccurs="0" maxOccurs="unbounded"/>  
2523       </xs:sequence>  
2524       <xs:attribute name="DataType" type="xs:anyURI"  
2525         use="required"/>  
2526       <xs:anyAttribute namespace="##any" processContents="lax"/>  
2527     </xs:extension>  
2528   </xs:complexContent>  
2529 </xs:complexType>
```

2530 The <AttributeValue> element is of AttributeValueType complex type.

2531 The <AttributeValue> element has the following attributes:

2532 DataType [Required]

2533 The data-type of the **attribute** value.

2534 5.32 Element <Obligations>

2535 The <Obligations> element SHALL contain a set of <Obligation> elements.

```
2536 <xs:element name="Obligations" type="xacml:ObligationsType"/>  
2537 <xs:complexType name="ObligationsType">  
2538   <xs:sequence>  
2539     <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>  
2540   </xs:sequence>  
2541 </xs:complexType>
```

2542 The <Obligations> element is of ObligationsType complexType.

2543 The <Obligations> element contains the following element:

2544 <Obligation> [One to Many]

2545 A sequence of **obligations**. See Section 5.34.

2546 5.33 Element <AssociatedAdvice>

2547 The <AssociatedAdvice> element SHALL contain a set of <Advice> elements.

```
2548 <xs:element name="AssociatedAdvice" type="xacml:AssociatedAdviceType"/>  
2549 <xs:complexType name="AssociatedAdviceType">  
2550   <xs:sequence>  
2551     <xs:element ref="xacml:Advice" maxOccurs="unbounded"/>  
2552   </xs:sequence>  
2553 </xs:complexType>
```

2554 The <AssociatedAdvice> element is of AssociatedAdviceType complexType.

2555 The <AssociatedAdvice> element contains the following element:
2556 <Advice> [One to Many]
2557 A sequence of **advice**. See Section 5.35.

2558 **5.34 Element <Obligation>**

2559 The <Obligation> element SHALL contain an identifier for the **obligation** and a set of **attributes** that
2560 form arguments of the action defined by the **obligation**.

```
2561 <xs:element name="Obligation" type="xacml:ObligationType"/>  
2562 <xs:complexType name="ObligationType">  
2563   <xs:sequence>  
2564     <xs:element ref="xacml:AttributeAssignment" minOccurs="0"  
2565       maxOccurs="unbounded"/>  
2566   </xs:sequence>  
2567   <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>  
2568 </xs:complexType>
```

2569 The <Obligation> element is of ObligationType complexType. See Section 7.18 for a description
2570 of how the set of **obligations** to be returned by the **PDP** is determined.

2571 The <Obligation> element contains the following elements and attributes:

2572 ObligationId [Required]

2573 **Obligation** identifier. The value of the **obligation** identifier SHALL be interpreted by the **PEP**.

2574 <AttributeAssignment> [Optional]

2575 **Obligation** arguments assignment. The values of the **obligation** arguments SHALL be
2576 interpreted by the **PEP**.

2577 **5.35 Element <Advice>**

2578 The <Advice> element SHALL contain an identifier for the **advice** and a set of **attributes** that form
2579 arguments of the supplemental information defined by the **advice**.

```
2580 <xs:element name="Advice" type="xacml:AdviceType"/>  
2581 <xs:complexType name="AdviceType">  
2582   <xs:sequence>  
2583     <xs:element ref="xacml:AttributeAssignment" minOccurs="0"  
2584       maxOccurs="unbounded"/>  
2585   </xs:sequence>  
2586   <xs:attribute name="AdviceId" type="xs:anyURI" use="required"/>  
2587 </xs:complexType>
```

2588 The <Advice> element is of AdviceType complexType. See Section 7.18 for a description of how the
2589 set of **advice** to be returned by the **PDP** is determined.

2590 The <Advice> element contains the following elements and attributes:

2591 AdviceId [Required]

2592 **Advice** identifier. The value of the **advice** identifier MAY be interpreted by the **PEP**.

2593 <AttributeAssignment> [Optional]

2594 **Advice** arguments assignment. The values of the **advice** arguments MAY be interpreted by the
2595 **PEP**.

2596 **5.36 Element <AttributeAssignment>**

2597 The <AttributeAssignment> element is used for including arguments in **obligation** and **advice**
2598 expressions. It SHALL contain an **AttributeId** and the corresponding **attribute** value, by extending
2599 the **AttributeValueType** type definition. The <AttributeAssignment> element MAY be used in

2600 any way that is consistent with the schema syntax, which is a sequence of `<xs:any>` elements. The
2601 value specified SHALL be understood by the *PEP*, but it is not further specified by XACML. See Section
2602 7.18. Section 4.2.4.3 provides a number of examples of arguments included in *obligation*-expressions.

```
2603 <xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
2604 <xs:complexType name="AttributeAssignmentType" mixed="true">
2605   <xs:complexContent>
2606     <xs:extension base="xacml:AttributeValueType">
2607       <xs:attribute name="AttributeId" type="xs:anyURI"
2608         use="required"/>
2609       <xs:attribute name="Category" type="xs:anyURI"
2610         use="optional"/>
2611       <xs:attribute name="Issuer" type="xs:string" use="optional"/>
2612     </xs:extension>
2613   </xs:complexContent>
2614 </xs:complexType>
```

2615 The `<AttributeAssignment>` element is of `AttributeAssignmentType` complex type.

2616 The `<AttributeAssignment>` element contains the following attributes:

2617 `AttributeId` [Required]

2618 The *attribute* Identifier.

2619 `Category` [Optional]

2620 An optional category of the *attribute*. If this attribute is missing, the *attribute* has no category.
2621 The *PEP* SHALL interpret the significance and meaning of any `Category` attribute. Non-
2622 normative note: an expected use of the category is to disambiguate *attributes* which are relayed
2623 from the request.

2624 `Issuer` [Optional]

2625 An optional issuer of the *attribute*. If this attribute is missing, the *attribute* has no issuer. The
2626 *PEP* SHALL interpret the significance and meaning of any `Issuer` attribute. Non-normative note:
2627 an expected use of the issuer is to disambiguate *attributes* which are relayed from the request.

2628 5.37 Element `<ObligationExpressions>`

2629 The `<ObligationExpressions>` element SHALL contain a set of `<ObligationExpression>`
2630 elements.

```
2631 <xs:element name="ObligationExpressions"
2632   type="xacml:ObligationExpressionsType"/>
2633 <xs:complexType name="ObligationExpressionsType">
2634   <xs:sequence>
2635     <xs:element ref="xacml:ObligationExpression" maxOccurs="unbounded"/>
2636   </xs:sequence>
2637 </xs:complexType>
```

2638 The `<ObligationExpressions>` element is of `ObligationExpressionsType` complexType.

2639 The `<ObligationExpressions>` element contains the following element:

2640 `<ObligationExpression>` [One to Many]

2641 A sequence of *obligations* expressions. See Section 5.39.

2642 5.38 Element `<AdviceExpressions>`

2643 The `<AdviceExpressions>` element SHALL contain a set of `<AdviceExpression>` elements.

```
2644 <xs:element name="AdviceExpressions" type="xacml:AdviceExpressionsType"/>
2645 <xs:complexType name="AdviceExpressionsType">
2646   <xs:sequence>
2647     <xs:element ref="xacml:AdviceExpression" maxOccurs="unbounded"/>
```

2648 </xs:sequence>
2649 </xs:complexType>

2650 The <AdviceExpressions> element is of AdviceExpressionsType complexType.

2651 The <AdviceExpressions> element contains the following element:

2652 <AdviceExpression> [One to Many]

2653 A sequence of **advice** expressions. See Section 5.40.

2654 5.39 Element <ObligationExpression>

2655 The <ObligationExpression> element evaluates to an **obligation** and SHALL contain an identifier

2656 for an **obligation** and a set of expressions that form arguments of the action defined by the **obligation**.

2657 The FulfillOn attribute SHALL indicate the **effect** for which this **obligation** must be fulfilled by the

2658 **PEP**.

```
2659 <xs:element name="ObligationExpression"  
2660     type="xacml:ObligationExpressionType"/>  
2661 <xs:complexType name="ObligationExpressionType">  
2662     <xs:sequence>  
2663         <xs:element ref="xacml:AttributeAssignmentExpression" minOccurs="0"  
2664             maxOccurs="unbounded"/>  
2665     </xs:sequence>  
2666     <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>  
2667     <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>  
2668 </xs:complexType>
```

2669 The <ObligationExpression> element is of ObligationExpressionType complexType. See

2670 Section 7.18 for a description of how the set of **obligations** to be returned by the **PDP** is determined.

2671 The <ObligationExpression> element contains the following elements and attributes:

2672 ObligationId [Required]

2673 **Obligation** identifier. The value of the **obligation** identifier SHALL be interpreted by the **PEP**.

2674 FulfillOn [Required]

2675 The **effect** for which this **obligation** must be fulfilled by the **PEP**.

2676 <AttributeAssignmentExpression> [Optional]

2677 **Obligation** arguments in the form of expressions. The expressions SHALL be evaluated by the

2678 PDP to constant <AttributeValue> elements or **bags**, which shall be the attribute

2679 assignments in the <Obligation> returned to the PEP. If an

2680 <AttributeAssignmentExpression> evaluates to an atomic **attribute** value, then there

2681 MUST be one resulting <AttributeAssignment> which MUST contain this single **attribute**

2682 value. If the <AttributeAssignmentExpression> evaluates to a **bag**, then there MUST be a

2683 resulting <AttributeAssignment> for each of the values in the **bag**. If the **bag** is empty, there

2684 shall be no <AttributeAssignment> from this <AttributeAssignmentExpression>.The

2685 values of the **obligation** arguments SHALL be interpreted by the **PEP**.

2686 5.40 Element <AdviceExpression>

2687 The <AdviceExpression> element evaluates to an **advice** and SHALL contain an identifier for an

2688 **advice** and a set of expressions that form arguments of the supplemental information defined by the

2689 **advice**. The AppliesTo attribute SHALL indicate the **effect** for which this **advice** must be provided to

2690 the **PEP**.

```
2691 <xs:element name="AdviceExpression" type="xacml:AdviceExpressionType"/>  
2692 <xs:complexType name="AdviceExpressionType">  
2693     <xs:sequence>
```

```

2694     <xs:element ref="xacml:AttributeAssignmentExpression" minOccurs="0"
2695     maxOccurs="unbounded"/>
2696   </xs:sequence>
2697   <xs:attribute name="AdviceId" type="xs:anyURI" use="required"/>
2698   <xs:attribute name="AppliesTo" type="xacml:EffectType" use="required"/>
2699 </xs:complexType>

```

2700 The <AdviceExpression> element is of AdviceExpressionType complexType. See Section 7.18
 2701 for a description of how the set of **advice** to be returned by the **PDP** is determined.

2702 The <AdviceExpression> element contains the following elements and attributes:

2703 AdviceId [Required]

2704 **Advice** identifier. The value of the **advice** identifier MAY be interpreted by the **PEP**.

2705 AppliesTo [Required]

2706 The **effect** for which this **advice** must be provided to the **PEP**.

2707 <AttributeAssignmentExpression> [Optional]

2708 **Advice** arguments in the form of expressions. The expressions SHALL be evaluated by the PDP
 2709 to constant <AttributeValue> elements or **bags**, which shall be the attribute assignments in
 2710 the <Advice> returned to the PEP. If an <AttributeAssignmentExpression> evaluates to
 2711 an atomic **attribute** value, then there MUST be one resulting <AttributeAssignment> which
 2712 MUST contain this single **attribute** value. If the <AttributeAssignmentExpression>
 2713 evaluates to a **bag**, then there MUST be a resulting <AttributeAssignment> for each of the
 2714 values in the **bag**. If the **bag** is empty, there shall be no <AttributeAssignment> from this
 2715 <AttributeAssignmentExpression>. The values of the **advice** arguments MAY be
 2716 interpreted by the **PEP**.

2717 5.41 Element <AttributeAssignmentExpression>

2718 The <AttributeAssignmentExpression> element is used for including arguments in **obligations**
 2719 and **advice**. It SHALL contain an AttributeId and an expression which SHALL be evaluated into the
 2720 corresponding **attribute** value. The value specified SHALL be understood by the **PEP**, but it is not further
 2721 specified by XACML. See Section 7.18. Section 4.2.4.3 provides a number of examples of arguments
 2722 included in **obligations**.

```

2723 <xs:element name="AttributeAssignmentExpression"
2724   type="xacml:AttributeAssignmentExpressionType"/>
2725 <xs:complexType name="AttributeAssignmentExpressionType">
2726   <xs:sequence>
2727     <xs:element ref="xacml:Expression"/>
2728   </xs:sequence>
2729   <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
2730   <xs:attribute name="Category" type="xs:anyURI" use="optional"/>
2731   <xs:attribute name="Issuer" type="xs:string" use="optional"/>
2732 </xs:complexType>

```

2733 The <AttributeAssignmentExpression> element is of AttributeAssignmentExpressionType
 2734 complex type.

2735 The <AttributeAssignmentExpression> element contains the following attributes:

2736 <Expression> [Required]

2737 The expression which evaluates to a constant **attribute** value or a bag of zero or more attribute
 2738 values. See section 5.25.

2739 AttributeId [Required]

2740 The **attribute** identifier. The value of the AttributeId attribute in the resulting
 2741 <AttributeAssignment> element MUST be equal to this value.

2742 Category [Optional]
2743 An optional category of the **attribute**. If this attribute is missing, the **attribute** has no category.
2744 The value of the `Category` attribute in the resulting `<AttributeAssignment>` element MUST be
2745 equal to this value.

2746 Issuer [Optional]
2747 An optional issuer of the **attribute**. If this attribute is missing, the **attribute** has no issuer. The
2748 value of the `Issuer` attribute in the resulting `<AttributeAssignment>` element MUST be equal to
2749 this value.

2750 5.42 Element `<Request>`

2751 The `<Request>` element is an abstraction layer used by the **policy** language. For simplicity of
2752 expression, this document describes **policy** evaluation in terms of operations on the **context**. However a
2753 conforming **PDP** is not required to actually instantiate the **context** in the form of an XML document. But,
2754 any system conforming to the XACML specification MUST produce exactly the same **authorization**
2755 **decisions** as if all the inputs had been transformed into the form of an `<Request>` element.

~~2756 The `<Request>` element contains `<Attributes>` elements. There may be multiple
2757 `<Attributes>` elements with the same `Category` attribute if the **PDP** implements the multiple
2758 decision profile, see [Multi]. Under other conditions, it is a syntax error if there are multiple
2759 `<Attributes>` elements with the same `Category` (see Section 7.19.2 for error codes).~~

```
2760 <xs:element name="Request" type="xacml:RequestType"/>  
2761 <xs:complexType name="RequestType">  
2762   <xs:sequence>  
2763     <xs:element ref="xacml:RequestDefaults" minOccurs="0"/>  
2764     <xs:element ref="xacml:Attributes" maxOccurs="unbounded"/>  
2765     <xs:element ref="xacml:MultiRequests" minOccurs="0"/>  
2766   </xs:sequence>  
2767   <xs:attribute name="ReturnPolicyIdList" type="xs:boolean" use="required"/>  
2768   <xs:attribute name="CombinedDecision" type="xs:boolean" use="required" />  
2769 </xs:complexType>
```

2770 The `<Request>` element is of `RequestType` complex type.

2771 The `<Request>` element contains the following elements and attributes:

2772 `ReturnPolicyIdList` [Required]

2773 This attribute is used to request that the **PDP** return a list of all fully applicable **policies** and
2774 **policy sets** which were used in the decision as a part of the decision response.

2775 `CombinedDecision` [Required]

2776 This attribute is used to request that the **PDP** combines multiple decisions into a single decision.
2777 The use of this attribute is specified in [Multi]. If the **PDP** does not implement the relevant
2778 functionality in [Multi], then the **PDP** must return an Indeterminate with a status code of
2779 `urn:oasis:names:tc:xacml:1.0:status:processing-error` if it receives a request with this attribute set
2780 to "true".

2781 `<RequestDefaults>` [Optional]

2782 Contains default values for the request, such as XPath version. See section 5.43.

2783 `<Attributes>` [One to Many]

2784 Specifies information about **attributes** of the request **context** by listing a sequence of
2785 `<Attribute>` elements associated with an **attribute** category. One or more `<Attributes>`
2786 elements are allowed. Different `<Attributes>` elements with different categories are used to
2787 represent information about the **subject**, **resource**, **action**, **environment** or other categories of
2788 the **access** request.

2789 The <Request> element contains <Attributes> elements. There may be multiple
2790 <Attributes> elements with the same Category attribute if the PDP implements the multiple
2791 decision profile, see [Multi]. Under other conditions, it is a syntax error if there are multiple
2792 <Attributes> elements with the same Category (see Section 7.19.2 for error codes).

2793 <MultiRequests> [Optional]

2794 Lists multiple **request contexts** by references to the <Attributes> elements. Implementation
2795 of this element is optional. The semantics of this element is defined in [Multi]. If the
2796 implementation does not implement this element, it MUST return an Indeterminate result if it
2797 encounters this element. See section 5.50.

2798 5.43 Element <RequestDefaults>

2799 The <RequestDefaults> element SHALL specify default values that apply to the <Request> element.

```
2800 <xs:element name="RequestDefaults" type="xacml:RequestDefaultsType"/>  
2801 <xs:complexType name="RequestDefaultsType">  
2802   <xs:sequence>  
2803     <xs:choice>  
2804       <xs:element ref="xacml:XPathVersion"/>  
2805     </xs:choice>  
2806   </xs:sequence>  
2807 </xs:complexType>
```

2808 <RequestDefaults> element is of RequestDefaultsType complex type.

2809 The <RequestDefaults> element contains the following elements:

2810 <XPathVersion> [Optional]

2811 Default XPath version for XPath expressions occurring in the request.

2812 5.44 Element <Attributes>

2813 The <Attributes> element specifies **attributes** of a **subject**, **resource**, **action**, **environment** or
2814 another category by listing a sequence of <Attribute> elements associated with the category.

```
2815 <xs:element name="Attributes" type="xacml:AttributesType"/>  
2816 <xs:complexType name="AttributesType">  
2817   <xs:sequence>  
2818     <xs:element ref="xacml:Content" minOccurs="0"/>  
2819     <xs:element ref="xacml:Attribute" minOccurs="0"  
2820       maxOccurs="unbounded"/>  
2821   </xs:sequence>  
2822   <xs:attribute name="Category" type="xs:anyURI" use="required"/>  
2823   <xs:attribute ref="xml:id" use="optional"/>  
2824 </xs:complexType><del><xs:complexType name="SubjectType"></del>
```

2825 The <Attributes> element is of AttributesType complex type.

2826 The <Attributes> element contains the following elements and attributes:

2827 Category [Required]

2828 This attribute indicates which **attribute** category the contained **attributes** belong to. The
2829 Category attribute is used to differentiate between **attributes** of **subject**, **resource**, **action**,
2830 **environment** or other categories.

2831 xml:id [Optional]

2832 This attribute provides a unique identifier for this <Attributes> element. See [XMLid] It is
2833 primarily intended to be referenced in multiple requests. See [Multi].

2834 <Content> [Optional]

2835 Specifies additional sources of **attributes** in free form XML document format which can be
2836 referenced using <AttributeSelector> elements.

2837 <Attribute> [Any Number]

2838 A sequence of **attributes** that apply to the category of the request.

2839 5.45 Element <Content>

2840 The <Content> element is a notional placeholder for additional **attributes**, typically the content of the
2841 **resource**.

```
2842 <xs:element name="Content" type="xacml:ContentType"/>  
2843 <xs:complexType name="ContentType" mixed="true">  
2844 <xs:sequence>  
2845 <xs:any namespace="##any" processContents="lax"/>  
2846 </xs:sequence>  
2847 </xs:complexType>
```

2848 The <Content> element is of ContentType complex type.

2849 The <Content> element has exactly one arbitrary type child element.

2850 5.46 Element <Attribute>

2851 The <Attribute> element is the central abstraction of the request **context**. It contains **attribute** meta-
2852 data and one or more **attribute** values. The **attribute** meta-data comprises the **attribute** identifier and
2853 the **attribute** issuer. <AttributeDesignator> elements in the **policy** MAY refer to **attributes** by
2854 means of this meta-data.

```
2855 <xs:element name="Attribute" type="xacml:AttributeType"/>  
2856 <xs:complexType name="AttributeType">  
2857 <xs:sequence>  
2858 <xs:element ref="xacml:AttributeValue" maxOccurs="unbounded"/>  
2859 </xs:sequence>  
2860 <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>  
2861 <xs:attribute name="Issuer" type="xs:string" use="optional"/>  
2862 <xs:attribute name="IncludeInResult" type="xs:boolean" use="required"/>  
2863 </xs:complexType>
```

2864 The <Attribute> element is of AttributeType complex type.

2865 The <Attribute> element contains the following attributes and elements:

2866 AttributeId [Required]

2867 The **Attribute** identifier. A number of identifiers are reserved by XACML to denote commonly
2868 used **attributes**. See Appendix Appendix B.

2869 Issuer [Optional]

2870 The **Attribute** issuer. For example, this attribute value MAY be an x500Name that binds to a
2871 public key, or it may be some other identifier exchanged out-of-band by issuing and relying
2872 parties.

2873 IncludeInResult [Default: false]

2874 Whether to include this **attribute** in the result. This is useful to correlate requests with their
2875 responses in case of multiple requests.

2876 <AttributeValue> [One to Many]

2877 One or more **attribute** values. Each **attribute** value MAY have contents that are empty, occur
2878 once or occur multiple times.

2879 5.47 Element <Response>

2880 The <Response> element is an abstraction layer used by the *policy* language. Any proprietary system
2881 using the XACML specification MUST transform an XACML *context* <Response> element into the form
2882 of its *authorization decision*.

2883 The <Response> element encapsulates the *authorization decision* produced by the *PDP*. It includes a
2884 sequence of one or more results, with one <Result> element per requested *resource*. Multiple results
2885 MAY be returned by some implementations, in particular those that support the XACML Profile for
2886 Requests for Multiple Resources [Multi]. Support for multiple results is OPTIONAL.

```
2887 <xs:element name="Response" type="xacml:ResponseType"/>  
2888 <xs:complexType name="ResponseType">  
2889   <xs:sequence>  
2890     <xs:element ref="xacml:Result" maxOccurs="unbounded"/>  
2891   </xs:sequence>  
2892 </xs:complexType>
```

2893 The <Response> element is of ResponseType complex type.

2894 The <Response> element contains the following elements:

2895 <Result> [One to Many]

2896 An *authorization decision* result. See Section 5.48.

2897 5.48 Element <Result>

2898 The <Result> element represents an *authorization decision* result. It MAY include a set of
2899 *obligations* that MUST be fulfilled by the *PEP*. If the *PEP* does not understand or cannot fulfill an
2900 *obligation*, then the action of the PEP is determined by its bias, see section 7.1. It MAY include a set of
2901 *advice* with supplemental information which MAY be safely ignored by the *PEP*.

```
2902 <xs:complexType name="ResultType">  
2903   <xs:sequence>  
2904     <xs:element ref="xacml:Decision"/>  
2905     <xs:element ref="xacml:Status" minOccurs="0"/>  
2906     <xs:element ref="xacml:Obligations" minOccurs="0"/>  
2907     <xs:element ref="xacml:AssociatedAdvice" minOccurs="0"/>  
2908     <xs:element ref="xacml:Attributes" minOccurs="0"  
2909       maxOccurs="unbounded"/>  
2910     <xs:element ref="xacml:PolicyIdentifierList" minOccurs="0"/>  
2911   </xs:sequence>  
2912 </xs:complexType>
```

2913 The <Result> element is of ResultType complex type.

2914 The <Result> element contains the following attributes and elements:

2915 <Decision> [Required]

2916 The *authorization decision*: "Permit", "Deny", "Indeterminate" or "NotApplicable".

2917 <Status> [Optional]

2918 Indicates whether errors occurred during evaluation of the *decision request*, and optionally,
2919 information about those errors. If the <Response> element contains <Result> elements whose
2920 <Status> elements are all identical, and the <Response> element is contained in a protocol
2921 wrapper that can convey status information, then the common status information MAY be placed
2922 in the protocol wrapper and this <Status> element MAY be omitted from all <Result>
2923 elements.

2924 <Obligations> [Optional]

2925 A list of *obligations* that MUST be fulfilled by the *PEP*. If the *PEP* does not understand or cannot
2926 fulfill an *obligation*, then the action of the PEP is determined by its bias, see section 7.2. See

2927 Section 7.18 for a description of how the set of **obligations** to be returned by the **PDP** is
2928 determined.

2929 <AssociatedAdvice> [Optional]

2930 A list of **advice** that provide supplemental information to the **PEP**. If the **PEP** does not
2931 understand an **advice**, the PEP may safely ignore the **advice**. See Section 7.18 for a description
2932 of how the set of **advice** to be returned by the **PDP** is determined.

2933 <Attributes> [Optional]

2934 A list of **attributes** that were part of the request. The choice of which **attributes** are included here
2935 is made with the `IncludeInResult` attribute of the <Attribute> elements of the request. See
2936 section 5.46.

2937 <PolicyIdentifierList> [Optional]

2938 If the `ReturnPolicyIdList` attribute in the <Request> is true (see section 5.42, 5.42), a **PDP**
2939 that implements this optional feature **MUST** return a list which includes the identifiers of all
2940 **policies** which were found to be fully applicable. That is, all **policies** where both the <Target>
2941 ~~matched and the <Condition> evaluated to true~~, whether or not the <Effect> was the same
2942 or different from the <Decision>. The list **MAY** include the identifiers of other policies which are
2943 currently in force, as long as no policies required for the decision are omitted. A **PDP** **MAY** satisfy
2944 this requirement by including all policies currently in force, or by including all policies which were
2945 evaluated in making the decision, or by including all policies which did not evaluate to
2946 "NotApplicable", or by any other algorithm which does not omit any policies which contributed to
2947 the decision. However, a decision which returns "NotApplicable" **MUST** return an empty list.
2948

2949 5.49 Element <PolicyIdentifierList>

2950 The <PolicyIdentifierList> element contains a list of **policy** and **policy set** identifiers of **policies**
2951 which have been applicable to a request. The list is unordered.

```
2952 <xs:element name="PolicyIdentifierList"  
2953   type="xacml:PolicyIdentifierListType"/>  
2954 <xs:complexType name="PolicyIdentifierListType">  
2955   <xs:choice minOccurs="0" maxOccurs="unbounded">  
2956     <xs:element ref="xacml:PolicyIdReference"/>  
2957     <xs:element ref="xacml:PolicySetIdReference"/>  
2958   </xs:choice>  
2959 </xs:complexType>
```

2960 The <PolicyIdentifierList> element is of `PolicyIdentifierListType` complex type.

2961 The <PolicyIdentifierList> element contains the following elements.

2962 <PolicyIdReference> [Any number]

2963 The identifier and version of a **policy** which was applicable to the request. See section 5.11. The
2964 <PolicyIdReference> element **MUST** use the `Version` attribute to specify the version and
2965 **MUST NOT** use the `LatestVersion` or `EarliestVersion` attributes.

2966 <PolicySetIdReference> [Any number]

2967 The identifier and version of a **policy set** which was applicable to the request. See section 5.10.
2968 The <PolicySetIdReference> element **MUST** use the `Version` attribute to specify the
2969 version and **MUST NOT** use the `LatestVersion` or `EarliestVersion` attributes.

2970 5.50 Element <MultiRequests>

2971 The <MultiRequests> element contains a list of requests by reference to <Attributes> elements in
2972 the enclosing <Request> element. The semantics of this element are defined in [Multi]. Support for this

2973 element is optional. If an implementation does not support this element, but receives it, the
2974 implementation MUST generate an “Indeterminate” response.

```
2975 <xs:element name="MultiRequests" type="xacml:MultiRequestsType"/>  
2976 <xs:complexType name="MultiRequestsType">  
2977   <xs:sequence>  
2978     <xs:element ref="xacml:RequestReference" maxOccurs="unbounded"/>  
2979   </xs:sequence>  
2980 </xs:complexType>
```

2981 The <MultiRequests> element contains the following elements.

2982 <RequestReference> [one to many]

2983 Defines a request instance by reference to <Attributes> elements in the enclosing
2984 <Request> element. See section 5.51.

2985 5.51 Element <RequestReference>

2986 The <RequestReference> element defines an instance of a request in terms of references to
2987 <Attributes> elements. The semantics of this element are defined in [Multi]. Support for this element
2988 is optional.

```
2989 <xs:element name="RequestReference" type="xacml:RequestReference"/>  
2990 <xs:complexType name="RequestReferenceType">  
2991   <xs:sequence>  
2992     <xs:element ref="xacml:AttributesReference" maxOccurs="unbounded"/>  
2993   </xs:sequence>  
2994 </xs:complexType>
```

2995 The <RequestReference> element contains the following elements.

2996 <AttributesReference> [one to many]

2997 A reference to an <Attributes> element in the enclosing <Request> element. See section
2998 5.52.

2999 5.52 Element <AttributesReference>

3000 The <AttributesReference> element makes a reference to an <Attributes> element. The
3001 meaning of this element is defined in [Multi]. Support for this element is optional.

```
3002 <xs:element name="AttributesReference" type="xacml:AttributesReference"/>  
3003 <xs:complexType name="AttributesReferenceType">  
3004   <xs:attribute name="ReferenceId" type="xs:IDREF" use="required"/>  
3005 </xs:complexType>
```

3006 The <AttributesReference> element contains the following attributes.

3007 ReferenceId [required]

3008 A reference to the xml:id attribute of an <Attributes> element in the enclosing <Request>
3009 element.

3010 5.53 Element <Decision>

3011 The <Decision> element contains the result of *policy* evaluation.

```
3012 <xs:element name="Decision" type="xacml:DecisionType"/>  
3013 <xs:simpleType name="DecisionType">  
3014   <xs:restriction base="xs:string">  
3015     <xs:enumeration value="Permit"/>  
3016     <xs:enumeration value="Deny"/>  
3017     <xs:enumeration value="Indeterminate"/>  
3018     <xs:enumeration value="NotApplicable"/>
```

3019 `</xs:restriction>`
3020 `</xs:simpleType>`

3021 The `<Decision>` element is of `DecisionType` simple type.

3022 The values of the `<Decision>` element have the following meanings:

3023 "Permit": the requested **access** is permitted.

3024 "Deny": the requested **access** is denied.

3025 "Indeterminate": the **PDP** is unable to evaluate the requested **access**. Reasons for such inability
3026 include: missing **attributes**, network errors while retrieving **policies**, division by zero during
3027 **policy** evaluation, syntax errors in the **decision request** or in the **policy**, etc.

3028 "NotApplicable": the **PDP** does not have any **policy** that applies to this **decision request**.

3029 **5.54 Element `<Status>`**

3030 The `<Status>` element represents the status of the **authorization decision** result.

```
3031 <xs:element name="Status" type="xacml:StatusType"/>
3032 <xs:complexType name="StatusType">
3033   <xs:sequence>
3034     <xs:element ref="xacml:StatusCode"/>
3035     <xs:element ref="xacml:StatusMessage" minOccurs="0"/>
3036     <xs:element ref="xacml:StatusDetail" minOccurs="0"/>
3037   </xs:sequence>
3038 </xs:complexType>
```

3039 The `<Status>` element is of `StatusType` complex type.

3040 The `<Status>` element contains the following elements:

3041 `<StatusCode>` [Required]

3042 Status code.

3043 `<StatusMessage>` [Optional]

3044 A status message describing the status code.

3045 `<StatusDetail>` [Optional]

3046 Additional status information.

3047 **5.55 Element `<StatusCode>`**

3048 The `<StatusCode>` element contains a major status code value and an optional [sequence](#)[recursive](#)
3049 [series](#) of minor status codes.

```
3050 <xs:element name="StatusCode" type="xacml:StatusCodeType"/>
3051 <xs:complexType name="StatusCodeType">
3052   <xs:sequence>
3053     <xs:element ref="xacml:StatusCode" minOccurs="0"/>
3054   </xs:sequence>
3055   <xs:attribute name="Value" type="xs:anyURI" use="required"/>
3056 </xs:complexType>
```

3057 The `<StatusCode>` element is of `StatusCodeType` complex type.

3058 The `<StatusCode>` element contains the following attributes and elements:

3059 Value [Required]

3060 See Section B.8 for a list of values.

3061 `<StatusCode>` [Any Number]

3062 Minor status code. This status code qualifies its parent status code.

3063 5.56 Element <StatusMessage>

3064 The <StatusMessage> element is a free-form description of the status code.

```
3065 <xs:element name="StatusMessage" type="xs:string"/>
```

3066 The <StatusMessage> element is of `xs:string` type.

3067 5.57 Element <StatusDetail>

3068 The <StatusDetail> element qualifies the <Status> element with additional information.

```
3069 <xs:element name="StatusDetail" type="xacml:StatusDetailType"/>
3070 <xs:complexType name="StatusDetailType">
3071   <xs:sequence>
3072     <xs:any namespace="##any" processContents="lax" minOccurs="0"
3073       maxOccurs="unbounded"/>
3074   </xs:sequence>
3075 </xs:complexType>
```

3076 The <StatusDetail> element is of `StatusDetailType` complex type.

3077 The <StatusDetail> element allows arbitrary XML content.

3078 Inclusion of a <StatusDetail> element is optional. However, if a **PDP** returns one of the following
3079 XACML-defined <StatusCode> values and includes a <StatusDetail> element, then the following
3080 rules apply.

3081 urn:oasis:names:tc:xacml:1.0:status:ok

3082 A **PDP** MUST NOT return a <StatusDetail> element in conjunction with the “ok” status value.

3083 urn:oasis:names:tc:xacml:1.0:status:missing-attribute

3084 A **PDP** MAY choose not to return any <StatusDetail> information or MAY choose to return a
3085 <StatusDetail> element containing one or more <MissingAttributeDetail> elements.

3086 urn:oasis:names:tc:xacml:1.0:status:syntax-error

3087 A **PDP** MUST NOT return a <StatusDetail> element in conjunction with the “syntax-error” status
3088 value. A syntax error may represent either a problem with the **policy** being used or with the request
3089 **context**. The **PDP** MAY return a <StatusMessage> describing the problem.

3090 urn:oasis:names:tc:xacml:1.0:status:processing-error

3091 A **PDP** MUST NOT return <StatusDetail> element in conjunction with the “processing-error” status
3092 value. This status code indicates an internal problem in the **PDP**. For security reasons, the **PDP** MAY
3093 choose to return no further information to the **PEP**. In the case of a divide-by-zero error or other
3094 computational error, the **PDP** MAY return a <StatusMessage> describing the nature of the error.

3095 5.58 Element <MissingAttributeDetail>

3096 The <MissingAttributeDetail> element conveys information about **attributes** required for **policy**
3097 evaluation that were missing from the request **context**.

```
3098 <xs:element name="MissingAttributeDetail"
3099   type="xacml:MissingAttributeDetailType"/>
3100 <xs:complexType name="MissingAttributeDetailType">
3101   <xs:sequence>
3102     <xs:element ref="xacml:AttributeValue" minOccurs="0"
3103       maxOccurs="unbounded"/>
3104   </xs:sequence>
3105   <xs:attribute name="Category" type="xs:anyURI" use="required"/>
3106   <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
3107   <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
3108   <xs:attribute name="Issuer" type="xs:string" use="optional"/>
```


3109 `</xs:complexType>`

3110 The `<MissingAttributeDetail>` element is of `MissingAttributeDetailType` complex type.

3111 The `<MissingAttributeDetail>` element contains the following attributes and elements:

3112 `<AttributeValue>` [Optional]

3113 The required value of the missing **attribute**.

3114 `Category` [Required]

3115 The category identifier of the missing **attribute**.

3116 `AttributeId` [Required]

3117 The identifier of the missing **attribute**.

3118 `DataType` [Required]

3119 The data-type of the missing **attribute**.

3120 `Issuer` [Optional]

3121 This attribute, if supplied, SHALL specify the required `Issuer` of the missing **attribute**.

3122 If the **PDP** includes `<AttributeValue>` elements in the `<MissingAttributeDetail>` element, then

3123 this indicates the acceptable values for that **attribute**. If no `<AttributeValue>` elements are included,

3124 then this indicates the names of **attributes** that the **PDP** failed to resolve during its evaluation. The list of

3125 **attributes** may be partial or complete. There is no guarantee by the **PDP** that supplying the missing

3126 values or **attributes** will be sufficient to satisfy the **policy**.

3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173

6 XPath 2.0 definitions

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

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

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

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

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

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

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

- 3174 StatusCode value SHALL be “urn:oasis:names:tc:xacml:1.0:status:processing-error”.
- 3175 Implementations MAY provide additional details about the error in the response or by some other
- 3176 means.
- 3177 3. For xs:decimal values the number of digits of precision returned by the numeric operators is
- 3178 implementation-defined.
- 3179 XACML leaves this implementation defined.
- 3180 4. If the number of digits in the result of a numeric operation exceeds the number of digits that the
- 3181 implementation supports, the result is truncated or rounded in an implementation-defined manner.
- 3182 XACML leaves this implementation defined.
- 3183 5. It is implementation-defined which version of Unicode is supported.
- 3184 XACML leaves this implementation defined. It is RECOMMENDED that the latest version is used.
- 3185 6. For fn:normalize-unicode, conforming implementations must support normalization form "NFC"
- 3186 and may support normalization forms "NFD", "NFKC", "NFKD", "FULLY-NORMALIZED". They
- 3187 may also support other normalization forms with implementation-defined semantics.
- 3188 XACML leaves this implementation defined.
- 3189 7. The ability to decompose strings into collation units suitable for substring matching is an
- 3190 implementation-defined property of a collation.
- 3191 XACML leaves this implementation defined.
- 3192 8. All minimally conforming processors must support year values with a minimum of 4 digits (i.e.,
- 3193 YYYY) and a minimum fractional second precision of 1 millisecond or three digits (i.e., s.sss).
- 3194 However, conforming processors may set larger implementation-defined limits on the maximum
- 3195 number of digits they support in these two situations.
- 3196 XACML leaves this implementation defined, and it is RECOMMENDED that users of XACML do
- 3197 not expect greater limits and precision.
- 3198 9. The result of casting a string to xs:decimal, when the resulting value is not too large or too small
- 3199 but nevertheless has too many decimal digits to be accurately represented, is implementation-
- 3200 defined.
- 3201 XACML leaves this implementation defined.
- 3202 10. Various aspects of the processing provided by fn:doc are implementation-defined.
- 3203 Implementations may provide external configuration options that allow any aspect of the
- 3204 processing to be controlled by the user.
- 3205 XACML leaves this implementation defined.
- 3206 11. The manner in which implementations provide options to weaken the stable characteristic of
- 3207 fn:collection and fn:doc are implementation-defined.
- 3208 XACML leaves this implementation defined.

3209 7 Functional requirements

3210 This section specifies certain functional requirements that are not directly associated with the production
3211 or consumption of a particular XACML element.

3212 Note that in each case an implementation is conformant as long as it produces the same result as is
3213 specified here, regardless of how and in what order the implementation behaves internally.

3214 7.1 Unicode issues

3215 7.1.1 Normalization

3216 In Unicode, some equivalent characters can be represented by more than one different Unicode
3217 character sequence. See [CMF]. The process of converting Unicode strings into equivalent character
3218 sequences is called "normalization" [UAX15]. Some operations, such as string comparison, are sensitive
3219 to normalization. An operation is normalization-sensitive if its output(s) are different depending on the
3220 state of normalization of the input(s); if the output(s) are textual, they are deemed different only if they
3221 would remain different were they to be normalized.

3222 For more information on normalization see [CM].

3223 An XACML implementation MUST behave as if each normalization-sensitive operation normalizes input
3224 strings into Unicode Normalization Form C ("NFC"). An implementation MAY use some other form of
3225 internal processing (such as using a non-Unicode, "legacy" character encoding) as long as the externally
3226 visible results are identical to this specification.

3227 7.1.2 Version of Unicode

3228 The version of Unicode used by XACML is implementation defined. It is RECOMMENDED that the latest
3229 version is used. Also note security issues in section 9.3.

3230 7.2 Policy enforcement point

3231 This section describes the requirements for the *PEP*.

3232 An application functions in the role of the *PEP* if it guards *access* to a set of *resources* and asks the
3233 *PDP* for an *authorization decision*. The *PEP* MUST abide by the *authorization decision* as described
3234 in one of the following sub-sections

3235 In any case any *advice* in the *decision* may be safely ignored by the *PEP*.

3236 7.2.1 Base PEP

3237 If the *decision* is "Permit", then the *PEP* SHALL permit *access*. If *obligations* accompany the *decision*,
3238 then the *PEP* SHALL permit *access* only if it understands and it can and will discharge those
3239 *obligations*.

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

3243 If the *decision* is "Indeterminate", then the *PEP*'s behavior is undefined.

3244 7.2.2 Deny-biased PEP

3245 If the *decision* is "Permit", then the *PEP* SHALL permit *access*. If *obligations* accompany the *decision*,
3246 then the *PEP* SHALL permit *access* only if it understands and it can and will discharge those
3247 *obligations*.

3248 All other **decisions** SHALL result in the denial of **access**.

3249 Note: other actions, e.g. consultation of additional **PDPs**, reformulation/resubmission of
3250 the **decision request**, etc., are not prohibited.

3251 7.2.3 Permit-biased PEP

3252 If the **decision** is "Deny", then the **PEP** SHALL deny **access**. If **obligations** accompany the **decision**,
3253 then the **PEP** shall deny **access** only if it understands, and it can and will discharge those **obligations**.

3254 All other **decisions** SHALL result in the permission of **access**.

3255 Note: other actions, e.g. consultation of additional **PDPs**, reformulation/resubmission of
3256 the **decision request**, etc., are not prohibited.

3257 7.3 Attribute evaluation

3258 **Attributes** are represented in the request **context** by the **context handler**, regardless of whether or not
3259 they appeared in the original **decision request**, and are referred to in the **policy** by attribute designators
3260 and attribute selectors. A **named attribute** is the term used for the criteria that the specific attribute
3261 designators use to refer to particular **attributes** in the <Attributes> elements of the request **context**.

3262 7.3.1 Structured attributes

3263 <AttributeValue> elements MAY contain an instance of a structured XML data-type, for example
3264 <ds:KeyInfo>. XACML 3.0 supports several ways for comparing the contents of such elements.

3265 1. In some cases, such elements MAY be compared using one of the XACML string functions, such
3266 as "string-regexp-match", described below. This requires that the element be given the data-type
3267 "http://www.w3.org/2001/XMLSchema#string". For example, a structured data-type that is
3268 actually a ds:KeyInfo/KeyName would appear in the **Context** as:

```
3269 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">  
3270 <ds:KeyName>jhibbert-key</ds:KeyName>  
3271 </AttributeValue>
```

3272 In general, this method will not be adequate unless the structured data-type is quite simple.

3273 2. The structured **attribute** MAY be made available in the <Content> element of the appropriate
3274 **attribute** category and an <AttributeSelector> element MAY be used to select the contents
3275 of a leaf sub-element of the structured data-type by means of an XPath expression. That value
3276 MAY then be compared using one of the supported XACML functions appropriate for its primitive
3277 data-type. This method requires support by the **PDP** for the optional XPath expressions feature.

3278 3. The structured **attribute** MAY be made available in the <Content> element of the appropriate
3279 **attribute** category and an <AttributeSelector> element MAY be used to select any node in
3280 the structured data-type by means of an XPath expression. This node MAY then be compared
3281 using one of the XPath-based functions described in Section A.3.15. This method requires
3282 support by the **PDP** for the optional XPath expressions and XPath functions features.

3283 See also Section 7.3.

3284 7.3.2 Attribute bags

3285 XACML defines implicit collections of its data-types. XACML refers to a collection of values that are of a
3286 single data-type as a **bag**. **Bags** of data-types are needed because selections of nodes from an XML
3287 **resource** or XACML request **context** may return more than one value.

3288 The <AttributeSelector> element uses an XPath expression to specify the selection of data from
3289 free form XML. The result of an XPath expression is termed a node-set, which contains all the nodes
3290 from the XML content that match the **predicate** in the XPath expression. Based on the various indexing
3291 functions provided in the XPath specification, it SHALL be implied that a resultant node-set is the

3292 collection of the matching nodes. XACML also defines the <AttributeDesignator> element to have
3293 the same matching methodology for **attributes** in the XACML request **context**.

3294 The values in a **bag** are not ordered, and some of the values may be duplicates. There SHALL be no
3295 notion of a **bag** containing **bags**, or a **bag** containing values of differing types; i.e., a **bag** in XACML
3296 SHALL contain only values that are of the same data-type.

3297 7.3.3 Multivalued attributes

3298 If a single <Attribute> element in a request **context** contains multiple <AttributeValue> child
3299 elements, then the **bag** of values resulting from evaluation of the <Attribute> element MUST be
3300 identical to the **bag** of values that results from evaluating a **context** in which each <AttributeValue>
3301 element appears in a separate <Attribute> element, each carrying identical meta-data.

3302 7.3.4 Attribute Matching

3303 A **named attribute** includes specific criteria with which to match **attributes** in the **context**. An **attribute**
3304 specifies a *Category*, *AttributeId* and *DataType*, and a **named attribute** also specifies the
3305 *Issuer*. A **named attribute** SHALL match an **attribute** if the values of their respective *Category*,
3306 *AttributeId*, *DataType* and optional *Issuer* attributes match. The *Category* of the **named**
3307 **attribute** MUST match, by **identifier equality**, the *Category* of the corresponding **context attribute**.
3308 The *AttributeId* of the **named attribute** MUST match, by **identifier equality**, the *AttributeId* of
3309 the corresponding **context attribute**. The *DataType* of the **named attribute** MUST match, by **identifier**
3310 **equality**, the *DataType* of the corresponding **context attribute**. If *Issuer* is supplied in the **named**
3311 **attribute**, then it MUST match, using the urn:oasis:names:tc:xacml:1.0:function:string-equal function, the
3312 *Issuer* of the corresponding **context attribute**. If *Issuer* is not supplied in the **named attribute**, then
3313 the matching of the **context attribute** to the **named attribute** SHALL be governed by *AttributeId* and
3314 *DataType* alone, regardless of the presence, absence, or actual value of *Issuer* in the corresponding
3315 **context attribute**. In the case of an attribute selector, the matching of the **attribute** to the **named**
3316 **attribute** SHALL be governed by the XPath expression and *DataType*.

3317 7.3.5 Attribute Retrieval

3318 The **PDP** SHALL request the values of **attributes** in the request **context** from the **context handler**. The
3319 **context handler** MAY also add **attributes** to the request **context** without the **PDP** requesting them. The
3320 **PDP** SHALL reference the **attributes** as if they were in a physical request **context** document, but the
3321 **context handler** is responsible for obtaining and supplying the requested values by whatever means it
3322 deems appropriate, including by retrieving them from one or more Policy Information Points. The **context**
3323 **handler** SHALL return the values of **attributes** that match the attribute designator or attribute selector
3324 and form them into a **bag** of values with the specified data-type. If no **attributes** from the request
3325 **context** match, then the **attribute** SHALL be considered missing. If the **attribute** is missing, then
3326 *MustBePresent* governs whether the attribute designator or attribute selector returns an empty **bag** or
3327 an “Indeterminate” result. If *MustBePresent* is “False” (default value), then a missing **attribute** SHALL
3328 result in an empty **bag**. If *MustBePresent* is “True”, then a missing **attribute** SHALL result in
3329 “Indeterminate”. This “Indeterminate” result SHALL be handled in accordance with the specification of the
3330 encompassing expressions, **rules**, **policies** and **policy sets**. If the result is “Indeterminate”, then the
3331 *AttributeId*, *DataType* and *Issuer* of the **attribute** MAY be listed in the **authorization decision** as
3332 described in Section 7.17. However, a **PDP** MAY choose not to return such information for security
3333 reasons.

3334 Regardless of any dynamic modifications of the request **context** during policy evaluation, the **PDP**
3335 SHALL behave as if each bag of **attribute** values is fully populated in the **context** before it is first tested,
3336 and is thereafter immutable during evaluation. (That is, every subsequent test of that **attribute** shall use
3337 the same bag of values that was initially tested.)

3338 **7.3.6 Environment Attributes**

3339 Standard *environment attributes* are listed in Section B.7. If a value for one of these *attributes* is
3340 supplied in the *decision request*, then the *context handler* SHALL use that value. Otherwise, the
3341 *context handler* SHALL supply a value. In the case of date and time *attributes*, the supplied value
3342 SHALL have the semantics of the "date and time that apply to the *decision request*".

3343 **7.3.7 AttributeSelector evaluation**

3344 An <AttributeSelector> element will be evaluated according to the following processing model.

3345

3346 NOTE: It is not necessary for an implementation to actually follow these steps. It is only
3347 necessary to produce results identical to those that would be produced by following these
3348 steps.

- 3349 1. Construct if the *attributes* category given by the *Category* attribute is not found or does not
3350 have a <Content> child element, then the return value is either "Indeterminate" or an empty *bag*
3351 as determined by the *MustBePresent* attribute; otherwise, construct an XML data structure
3352 suitable for xpath processing from the <Content> element in the attributes category given by the
3353 *Category* attribute.- The data structure shall be constructed so that the document node of this
3354 structure contains a single document element which corresponds to the single child element of
3355 the <Content> element. The constructed data structure shall be equivalent to one that would
3356 result from parsing a stand-alone XML document consisting of the contents of the <Content>
3357 element (including any comment and processing-instruction markup). Namespace declarations
3358 which are not "visibly utilized", as defined by [exc-c14n], MAY not be present and MUST NOT be
3359 utilized by the XPath expression in step 3. The data structure must meet the requirements of the
3360 applicable xpath version.
- 3361 2. Select a context node for xpath processing from this data structure. If there is a
3362 *ContextSelectorId* attribute, the context node shall be the node selected by applying the
3363 XPath expression given in the *attribute* value of the designated *attribute* (in the *attributes*
3364 category given by the <AttributeSelector> *Category* attribute). It shall be an error if this
3365 evaluation returns no node or more than one node, in which case the return value MUST be an
3366 "Indeterminate" with a status code "urn:oasis:names:tc:xacml:1.0:status:syntax-error". If there is
3367 no *ContextSelectorId*, the document node of the data structure shall be the context node.
- 3368 3. Evaluate the XPath expression given in the *Path* attribute against the xml data structure, using
3369 the context node selected in the previous step. It shall be an error if this evaluation returns
3370 anything other than a sequence of nodes (possibly empty), in which case the
3371 <AttributeSelector> MUST return "Indeterminate" with a status code
3372 "urn:oasis:names:tc:xacml:1.0:status:syntax-error". If the evaluation returns an empty sequence
3373 of nodes, then the return value is either "Indeterminate" or an empty *bag* as determined by the
3374 *MustBePresent* attribute.
- 3375 4. If the data type is a primitive data type, convert the text value of each selected node to the
3376 desired data type, as specified in the *DataType* attribute. Each value shall be constructed using
3377 the appropriate constructor function from [XF] Section 5 listed below, corresponding to the
3378 specified data type.

- 3379
3380 xs:string()
3381 xs:boolean()
3382 xs:integer()
3383 xs:double()
3384 xs:dateTime()
3385 xs:date()
3386 xs:time()
3387 xs:hexBinary()
3388 xs:base64Binary()

3389 xs:anyURI()
3390 xs:yearMonthDuration()
3391 xs:dayTimeDuration()
3392

3393 If the `DataType` is not one of the primitive types listed above, then the return values shall be
3394 constructed from the nodeset in a manner specified by the particular `DataType` extension
3395 specification. If the data type extension does not specify an appropriate constructor function, then
3396 the `<AttributeSelector>` MUST return "Indeterminate" with a status code
3397 "urn:oasis:names:tc:xacml:1.0:status:syntax-error".
3398

3399 If an error occurs when converting the values returned by the XPath expression to the specified
3400 `DataType`, then the result of the `<AttributeSelector>` MUST be "Indeterminate", with a
3401 status code "urn:oasis:names:tc:xacml:1.0:status:processing-error"

3402 7.4 Expression evaluation

3403 XACML specifies expressions in terms of the elements listed below, of which the `<Apply>` and
3404 `<Condition>` elements recursively compose greater expressions. Valid expressions SHALL be type
3405 correct, which means that the types of each of the elements contained within `<Apply>` elements SHALL
3406 agree with the respective argument types of the function that is named by the `FunctionId` attribute.
3407 The resultant type of the `<Apply>` element SHALL be the resultant type of the function, which MAY be
3408 narrowed to a primitive data-type, or a **bag** of a primitive data-type, by type-unification. XACML defines
3409 an evaluation result of "Indeterminate", which is said to be the result of an invalid expression, or an
3410 operational error occurring during the evaluation of the expression.

3411 XACML defines these elements to be in the substitution group of the `<Expression>` element:

- 3412 • `<xacml:AttributeValue>`
- 3413 • `<xacml:AttributeDesignator>`
- 3414 • `<xacml:AttributeSelector>`
- 3415 • `<xacml:Apply>`
- 3416 • `<xacml:Function>`
- 3417 • `<xacml:VariableReference>`

3418 7.5 Arithmetic evaluation

3419 IEEE 754 [IEEE754] specifies how to evaluate arithmetic functions in a context, which specifies defaults
3420 for precision, rounding, etc. XACML SHALL use this specification for the evaluation of all integer and
3421 double functions relying on the Extended Default Context, enhanced with double precision:

3422 flags - all set to 0

3423 trap-enablers - all set to 0 (IEEE 854 §7) with the exception of the "division-by-zero" trap enabler,
3424 which SHALL be set to 1

3425 precision - is set to the designated double precision

3426 rounding - is set to round-half-even (IEEE 854 §4.1)

3427 7.6 Match evaluation

3428 The **attribute** matching element `<Match>` appears in the `<Target>` element of **rules**, **policies** and
3429 **policy sets**.

3430 This element represents a Boolean expression over **attributes** of the request **context**. A matching
3431 element contains a `MatchId` attribute that specifies the function to be used in performing the match
3432 evaluation, an `<AttributeValue>` and an `<AttributeDesignator>` or `<AttributeSelector>`
3433 element that specifies the **attribute** in the **context** that is to be matched against the specified value.

3434 The `MatchId` attribute SHALL specify a function that takes two arguments, returning a result type of
3435 "http://www.w3.org/2001/XMLSchema#boolean". The **attribute** value specified in the matching element
3436 SHALL be supplied to the `MatchId` function as its first argument. An element of the **bag** returned by the
3437 `<AttributeDesignator>` or `<AttributeSelector>` element SHALL be supplied to the `MatchId`
3438 function as its second argument, as explained below. The `DataType` of the `<AttributeValue>`
3439 SHALL match the data-type of the first argument expected by the `MatchId` function. The `DataType` of
3440 the `<AttributeDesignator>` or `<AttributeSelector>` element SHALL match the data-type of the
3441 second argument expected by the `MatchId` function.

3442 In addition, functions that are strictly within an extension to XACML MAY appear as a value for the
3443 `MatchId` attribute, and those functions MAY use data-types that are also extensions, so long as the
3444 extension function returns a Boolean result and takes two single base types as its inputs. The function
3445 used as the value for the `MatchId` attribute SHOULD be easily indexable. Use of non-indexable or
3446 complex functions may prevent efficient evaluation of **decision requests**.

3447 The evaluation semantics for a matching element is as follows. If an operational error were to occur while
3448 evaluating the `<AttributeDesignator>` or `<AttributeSelector>` element, then the result of the
3449 entire expression SHALL be "Indeterminate". If the `<AttributeDesignator>` or
3450 `<AttributeSelector>` element were to evaluate to an empty **bag**, then the result of the expression
3451 SHALL be "False". Otherwise, the `MatchId` function SHALL be applied between the
3452 `<AttributeValue>` and each element of the **bag** returned from the `<AttributeDesignator>` or
3453 `<AttributeSelector>` element. If at least one of those function applications were to evaluate to
3454 "True", then the result of the entire expression SHALL be "True". Otherwise, if at least one of the function
3455 applications results in "Indeterminate", then the result SHALL be "Indeterminate". Finally, if all function
3456 applications evaluate to "False", then the result of the entire expression SHALL be "False".

3457 It is also possible to express the semantics of a **target** matching element in a **condition**. For instance,
3458 the **target** match expression that compares a "**subject-name**" starting with the name "John" can be
3459 expressed as follows:

```
3460 <Match
3461 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
3462   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
3463     John.*
3464   </AttributeValue>
3465   <AttributeDesignator
3466     Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
3467 subject"
3468     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
3469     DataType="http://www.w3.org/2001/XMLSchema#string"/>
3470 </Match>
```

3471 Alternatively, the same match semantics can be expressed as an `<Apply>` element in a **condition** by
3472 using the "urn:oasis:names:tc:xacml:3.0:function:any-of" function, as follows:

```
3473 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">
3474   <Function
3475   FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match"/>
3476   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
3477     John.*
3478   </AttributeValue>
3479   <AttributeDesignator
3480     Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
3481 subject"
3482     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
3483     DataType="http://www.w3.org/2001/XMLSchema#string"/>
3484 </Apply>
```

3485 **7.7 Target evaluation**

3486 An empty **target** matches any request. Otherwise the **target** value SHALL be "Match" if all the AnyOf
 3487 specified in the **target** match values in the request **context**. Otherwise, if any one of the AnyOf specified
 3488 in the **target** is "No Match", then the **target** SHALL be "No Match". Otherwise, the **target** SHALL be
 3489 "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"

3490 *Table 1 Target match table*

3491 The AnyOf SHALL match values in the request **context** if at least one of their <AllOf> elements
 3492 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"

3493 *Table 2 AnyOf match table*

3494 An AllOf SHALL match a value in the request **context** if the value of all its <Match> elements is "True".

3495 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"

3496 *Table 3 AllOf match table*

3497 **7.8 VariableReference Evaluation**

3498 The <VariableReference> element references a single <VariableDefinition> element contained
 3499 within the same <Policy> element. A <VariableReference> that does not reference a particular
 3500 <VariableDefinition> element within the encompassing <Policy> element is called an undefined
 3501 reference. **Policies** with undefined references are invalid.

3502 In any place where a <VariableReference> occurs, it has the effect as if the text of the
 3503 <Expression> element defined in the <VariableDefinition> element replaces the
 3504 <VariableReference> element. Any evaluation scheme that preserves this semantic is acceptable.
 3505 For instance, the expression in the <VariableDefinition> element may be evaluated to a particular
 3506 value and cached for multiple references without consequence. (I.e. the value of an <Expression>
 3507 element remains the same for the entire **policy** evaluation.) This characteristic is one of the benefits of
 3508 XACML being a declarative language.

3509 A variable reference containing circular references is invalid. The PDP MUST detect circular references
 3510 either at policy loading time or during runtime evaluation. If the PDP detects a circular reference during

3511 runtime the variable reference evaluates to “Indeterminate” with status code
 3512 urn:oasis:names:tc:xacml:1.0:status:processing-error.

3513 7.9 Condition evaluation

3514 The **condition** value SHALL be "True" if the <Condition> element is absent, or if it evaluates to "True".
 3515 Its value SHALL be "False" if the <Condition> element evaluates to "False". The **condition** value
 3516 SHALL be "Indeterminate", if the expression contained in the <Condition> element evaluates to
 3517 "Indeterminate."

3518 7.10 Extended Indeterminate

3519 Some **combining algorithms** are defined in terms of an extended set of “Indeterminate” values. The
 3520 extended set associated with the “Indeterminate” contains the potential effect values which could have
 3521 occurred if there would not have been an error causing the “Indeterminate”. The possible extended set
 3522 “Indeterminate” values are

- 3523 • “Indeterminate{D}”: an “Indeterminate” from a **policy** or **rule** which could have evaluated to “Deny”,
 3524 but not “Permit”
- 3525 • “Indeterminate{P}”: an “Indeterminate” from a **policy** or **rule** which could have evaluated to “Permit”,
 3526 but not “Deny”
- 3527 • “Indeterminate{DP}”: an “Indeterminate” from a **policy** or **rule** which could have evaluated to “Deny”
 3528 or “Permit”.

3529 The **combining algorithms** which are defined in terms of the extended “Indeterminate” make use of the
 3530 additional information to allow for better treatment of errors in the algorithms.

3531 The final decision returned by a **PDP** cannot be an extended Indeterminate. Any such decision at the top
 3532 level **policy** or **policy set** is returned as a plain Indeterminate in the response from the **PDP**.

3533 The tables in the following four sections define how extended “Indeterminate” values are produced during
 3534 **Rule**, **Policy** and **PolicySet** evaluation.

3535 7.11 Rule evaluation

3536 A **rule** has a value that can be calculated by evaluating its contents. **Rule** evaluation involves separate
 3537 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

3538 Table 4 Rule truth table.

3539 7.12 Policy evaluation

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

3543 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

3544 Table 5 Policy truth table

3545 Note that none of the **rule-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 **rules**, MUST be taken into account when evaluating the **policy**. The
 3548 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 7.13 Policy Set evaluation

3552 The value of a **policy set** SHALL be determined by its contents, considered in relation to the contents of
 3553 the request **context**. A **policy set**'s value SHALL be determined by evaluation of the **policy set**'s **target**,
 3554 and, according to the specified **policy-combining algorithm**, **policies** and **policy sets**,

3555 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

3556 Table 6 Policy set truth table

3557 Note that none of the **policy-combining algorithms** defined by XACML 3.0 take parameters. However,
 3558 non-standard combining algorithms MAY take parameters. In such a case, the values of these
 3559 parameters associated with the **policies**, MUST be taken into account when evaluating the **policy set**.
 3560 The parameters and their types should be defined in the specification of the combining algorithm. If the
 3561 implementation supports combiner parameters and if combiner parameters are present in a **policy**, then
 3562 the parameter values MUST be supplied to the combining algorithm implementation.

3563 7.14 Policy and Policy set value for Indeterminate Target

3564 If the **target** of a **policy** or **policy set** evaluates to “Indeterminate”, the value of the **policy** or **policy set**
 3565 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}"

3566 Table 7 The value of a **policy** or **policy set** when the target is "Indeterminate".

3567 7.15 PolicySetIdReference and PolicyIdReference evaluation

3568 A policy set id reference or a policy id reference is evaluated by resolving the reference and evaluating
3569 the referenced policy set or policy.

3570 If resolving the reference fails, the reference evaluates to "Indeterminate" with status code
3571 urn:oasis:names:tc:xacml:1.0:status:processing-error.

3572 A policy set id reference or a policy id reference containing circular references is invalid. The PDP MUST
3573 detect circular references either at policy loading time or during runtime evaluation. If the PDP detects a
3574 circular reference during runtime the reference evaluates to "Indeterminate" with status code
3575 urn:oasis:names:tc:xacml:1.0:status:processing-error.

3576 7.16 Hierarchical resources

3577 It is often the case that a **resource** is organized as a hierarchy (e.g. file system, XML document). XACML
3578 provides several optional mechanisms for supporting hierarchical **resources**. These are described in the
3579 XACML Profile for Hierarchical Resources [**Hier**] and in the XACML Profile for Requests for Multiple
3580 Resources [**Multi**].

3581 7.17 Authorization decision

3582 In relation to a particular **decision request**, the **PDP** is defined by a **policy-combining algorithm** and a
3583 set of **policies** and/or **policy sets**. The **PDP** SHALL return a response **context** as if it had evaluated a
3584 single **policy set** consisting of this **policy-combining algorithm** and the set of **policies** and/or **policy**
3585 **sets**.

3586 The **PDP** MUST evaluate the **policy set** as specified in Sections 5 and 7. The **PDP** MUST return a
3587 response **context**, with one <Decision> element of value "Permit", "Deny", "Indeterminate" or
3588 "NotApplicable".

3589 If the **PDP** cannot make a **decision**, then an "Indeterminate" <Decision> element SHALL be returned.

3590 7.18 Obligations and advice

3591 A **rule**, **policy**, or **policy set** may contain one or more **obligation** or **advice** expressions. When such a
3592 **rule**, **policy**, or **policy set** is evaluated, the **obligation** or **advice** expression SHALL be evaluated to an
3593 **obligation** or **advice** respectively, which SHALL be passed up to the next level of evaluation (the
3594 enclosing or referencing **policy**, **policy set**, or **authorization decision**) only if the result of the **rule**,
3595 **policy**, or **policy set** being evaluated matches the value of the FulfillOn attribute of the **obligation** or
3596 the AppliesTo attribute of the **advice**. If any of the **attribute** assignment expressions in an **obligation**
3597 or **advice** expression with a matching FulfillOn or AppliesTo attribute evaluates to "Indeterminate",
3598 then the whole **rule**, **policy**, or **policy set** SHALL be "Indeterminate". If the FulfillOn or AppliesTo
3599 attribute does not match the result of the combining algorithm or the **rule** evaluation, then any
3600 indeterminate in an **obligation** or **advice** expression has no effect.

3601 As a consequence of this procedure, no **obligations** or **advice** SHALL be returned to the **PEP** if the **rule**,
3602 **policies**, or **policy sets** from which they are drawn are not evaluated, or if their evaluated result is
3603 "Indeterminate" or "NotApplicable", or if the **decision** resulting from evaluating the **rule**, **policy**, or **policy**
3604 **set** does not match the **decision** resulting from evaluating an enclosing **policy set**.

3605 If the **PDP**'s evaluation is viewed as a tree of **rules**, **policy sets** and **policies**, each of which returns
3606 "Permit" or "Deny", then the set of **obligations** and **advice** returned by the **PDP** to the **PEP** will include
3607 only the **obligations** and **advice** associated with those paths where the result at each level of evaluation
3608 is the same as the result being returned by the **PDP**. In situations where any lack of determinism is
3609 unacceptable, a deterministic combining algorithm, such as ordered-deny-overrides, should be used.

3610 Also see Section 7.2.

3611 7.19 Exception handling

3612 XACML specifies behavior for the **PDP** in the following situations.

3613 7.19.1 Unsupported functionality

3614 If the **PDP** attempts to evaluate a **policy set** or **policy** that contains an optional element type or function
3615 that the **PDP** does not support, then the **PDP** SHALL return a <Decision> value of "Indeterminate". If a
3616 <StatusCode> element is also returned, then its value SHALL be
3617 "urn:oasis:names:tc:xacml:1.0:status:syntax-error" in the case of an unsupported element type, and
3618 "urn:oasis:names:tc:xacml:1.0:status:processing-error" in the case of an unsupported function.

3619 7.19.2 Syntax and type errors

3620 If a **policy** that contains invalid syntax is evaluated by the XACML **PDP** at the time a **decision request** is
3621 received, then the result of that **policy** SHALL be "Indeterminate" with a `StatusCode` value of
3622 "urn:oasis:names:tc:xacml:1.0:status:syntax-error".

3623 If a **policy** that contains invalid static data-types is evaluated by the XACML **PDP** at the time a **decision**
3624 **request** is received, then the result of that **policy** SHALL be "Indeterminate" with a `StatusCode` value of
3625 "urn:oasis:names:tc:xacml:1.0:status:processing-error".

3626 7.19.3 Missing attributes

3627 The absence of matching **attributes** in the request **context** for any of the attribute designators attribute or
3628 selectors that are found in the **policy** will result in an enclosing <AllOf> element to return a value of
3629 "Indeterminate", if the designator or selector has the `MustBePresent` XML attribute set to true, as
3630 described in Sections 5.29 and 5.30 and may result in a <Decision> element containing the
3631 "Indeterminate" value. If, in this case a status code is supplied, then the value

3632 "urn:oasis:names:tc:xacml:1.0:status:missing-attribute"

3633 SHALL be used, to indicate that more information is needed in order for a definitive **decision** to be
3634 rendered. In this case, the <Status> element MAY list the names and data-types of any **attributes** that
3635 are needed by the **PDP** to refine its **decision** (see Section 5.58). A **PEP** MAY resubmit a refined request
3636 **context** in response to a <Decision> element contents of "Indeterminate" with a status code of

3637 "urn:oasis:names:tc:xacml:1.0:status:missing-attribute"

3638 by adding **attribute** values for the **attribute** names that were listed in the previous response. When the
3639 **PDP** returns a <Decision> element contents of "Indeterminate", with a status code of

3640 "urn:oasis:names:tc:xacml:1.0:status:missing-attribute",

3641 it MUST NOT list the names and data-types of any **attribute** for which values were supplied in the original
3642 request. Note, this requirement forces the **PDP** to eventually return an **authorization decision** of
3643 "Permit", "Deny", or "Indeterminate" with some other status code, in response to successively-refined
3644 requests.

3645 7.20 Identifier equality

3646 XACML makes use of URIs and strings as identifiers. When such identifiers are compared for equality,
3647 the comparison MUST be done so that the identifiers are equal if they have the same length and the
3648 characters in the two identifiers are equal codepoint by codepoint.

3649 The following is a list of the identifiers which MUST use this definition of equality.

3650 The content of the element <XPathVersion>.

3651 The XML attribute `Value` in the element <StatusCode>.

- 3652 The XML attributes `Category`, `AttributeId`, `DataType` and `Issuer` in the element
3653 `<MissingAttributeDetail>`.
- 3654 The XML attribute `Category` in the element `<Attributes>`.
- 3655 The XML attributes `AttributeId` and `Issuer` in the element `<Attribute>`.
- 3656 The XML attribute `ObligationId` in the element `<Obligation>`.
- 3657 The XML attribute `AdviceId` in the element `<Advice>`.
- 3658 The XML attributes `AttributeId` and `Category` in the element `<AttributeAssignment>`.
- 3659 The XML attribute `ObligationId` in the element `<ObligationExpression>`.
- 3660 The XML attribute `AdviceId` in the element `<AdviceExpression>`.
- 3661 The XML attributes `AttributeId`, `Category` and `Issuer` in the element
3662 `<AttributeAssignmentExpression>`.
- 3663 The XML attributes `PolicySetId` and `PolicyCombiningAlgId` in the element `<PolicySet>`.
- 3664 The XML attribute `ParameterName` in the element `<CombinerParameter>`.
- 3665 The XML attribute `RuleIdRef` in the element `<RuleCombinerParameters>`.
- 3666 The XML attribute `PolicyIdRef` in the element `<PolicyCombinerParameters>`.
- 3667 The XML attribute `PolicySetIdRef` in the element `<PolicySetCombinerParameters>`.
- 3668 The anyURI in the content of the complex type `IdReferenceType`.
- 3669 The XML attributes `PolicyId` and `RuleCombiningAlgId` in the element `<Policy>`.
- 3670 The XML attribute `RuleId` in the element `<Rule>`.
- 3671 The XML attribute `MatchId` in the element `<Match>`.
- 3672 The XML attribute `VariableId` in the element `<VariableDefinition>`.
- 3673 The XML attribute `VariableId` in the element `<VariableReference>`.
- 3674 The XML attributes `Category`, `ContextSelectorId` and `DataType` in the element
3675 `<AttributeSelector>`.
- 3676 The XML attributes `Category`, `AttributeId`, `DataType` and `Issuer` in the element
3677 `<AttributeDesignator>`.
- 3678 The XML attribute `DataType` in the element `<AttributeValue>`.
- 3679 The XML attribute `FunctionId` in the element `<Function>`.
- 3680 The XML attribute `FunctionId` in the element `<Apply>`.
- 3681
- 3682 It is RECOMMENDED that extensions to XACML use the same definition of identifier equality for similar
3683 identifiers.
- 3684 It is RECOMMENDED that extensions which define identifiers do not define identifiers which could be
3685 easily misinterpreted by people as being subject to other kind of processing, such as URL character
3686 escaping, before matching.

3687 8 XACML extensibility points (non-normative)

3688 This section describes the points within the XACML model and schema where extensions can be added.

3689 8.1 Extensible XML attribute types

3690 The following XML attributes have values that are URIs. These may be extended by the creation of new
3691 URIs associated with new semantics for these attributes.

3692 `Category`,

3693 `AttributeId`,

3694 `DataType`,

3695 `FunctionId`,

3696 `MatchId`,

3697 `ObligationId`,

3698 `AdviceId`,

3699 `PolicyCombiningAlgId`,

3700 `RuleCombiningAlgId`,

3701 `StatusCodes`.

3702 ~~`SubjectCategory`.~~

3703 See Section 5 for definitions of these *attribute* types.

3704 8.2 Structured attributes

3705 <AttributeValue> elements MAY contain an instance of a structured XML data-type. Section 7.3.1
3706 describes a number of standard techniques to identify data items within such a structured *attribute*.
3707 Listed here are some additional techniques that require XACML extensions.

- 3708 1. For a given structured data-type, a community of XACML users MAY define new *attribute*
3709 identifiers for each leaf sub-element of the structured data-type that has a type conformant with
3710 one of the XACML-defined primitive data-types. Using these new *attribute* identifiers, the *PEPs*
3711 or *context handlers* used by that community of users can flatten instances of the structured
3712 data-type into a sequence of individual <Attribute> elements. Each such <Attribute>
3713 element can be compared using the XACML-defined functions. Using this method, the structured
3714 data-type itself never appears in an <AttributeValue> element.
- 3715 2. A community of XACML users MAY define a new function that can be used to compare a value of
3716 the structured data-type against some other value. This method may only be used by *PDPs* that
3717 support the new function.

3718 9 Security and privacy considerations (non- 3719 normative)

3720 This section identifies possible security and privacy compromise scenarios that should be considered
3721 when implementing an XACML-based system. The section is informative only. It is left to the
3722 implementer to decide whether these compromise scenarios are practical in their environment and to
3723 select appropriate safeguards.

3724 9.1 Threat model

3725 We assume here that the adversary has access to the communication channel between the XACML
3726 actors and is able to interpret, insert, delete, and modify messages or parts of messages.

3727 Additionally, an actor may use information from a former message maliciously in subsequent transactions.
3728 It is further assumed that *rules* and *policies* are only as reliable as the actors that create and use them.
3729 Thus it is incumbent on each actor to establish appropriate trust in the other actors upon which it relies.
3730 Mechanisms for trust establishment are outside the scope of this specification.

3731 The messages that are transmitted between the actors in the XACML model are susceptible to attack by
3732 malicious third parties. Other points of vulnerability include the *PEP*, the *PDP*, and the *PAP*. While some
3733 of these entities are not strictly within the scope of this specification, their compromise could lead to the
3734 compromise of *access control* enforced by the *PEP*.

3735 It should be noted that there are other components of a distributed system that may be compromised,
3736 such as an operating system and the domain-name system (DNS) that are outside the scope of this
3737 discussion of threat models. Compromise in these components may also lead to a policy violation.

3738 The following sections detail specific compromise scenarios that may be relevant to an XACML system.

3739 9.1.1 Unauthorized disclosure

3740 XACML does not specify any inherent mechanisms to protect the confidentiality of the messages
3741 exchanged between actors. Therefore, an adversary could observe the messages in transit. Under
3742 certain security *policies*, disclosure of this information is a violation. Disclosure of *attributes* or the types
3743 of *decision requests* that a *subject* submits may be a breach of privacy policy. In the commercial
3744 sector, the consequences of unauthorized disclosure of personal data may range from embarrassment to
3745 the custodian, to imprisonment and/or large fines in the case of medical or financial data.

3746 Unauthorized disclosure is addressed by confidentiality safeguards.

3747 9.1.2 Message replay

3748 A message replay attack is one in which the adversary records and replays legitimate messages between
3749 XACML actors. This attack may lead to denial of service, the use of out-of-date information or
3750 impersonation.

3751 Prevention of replay attacks requires the use of message freshness safeguards.

3752 Note that encryption of the message does not mitigate a replay attack since the message is simply
3753 replayed and does not have to be understood by the adversary.

3754 9.1.3 Message insertion

3755 A message insertion attack is one in which the adversary inserts messages in the sequence of messages
3756 between XACML actors.

3757 The solution to a message insertion attack is to use mutual authentication and message sequence
3758 integrity safeguards between the actors. It should be noted that just using SSL mutual authentication is
3759 not sufficient. This only proves that the other party is the one identified by the *subject* of the X.509

3760 certificate. In order to be effective, it is necessary to confirm that the certificate **subject** is authorized to
3761 send the message.

3762 9.1.4 Message deletion

3763 A message deletion attack is one in which the adversary deletes messages in the sequence of messages
3764 between XACML actors. Message deletion may lead to denial of service. However, a properly designed
3765 XACML system should not render an incorrect **authorization decision** as a result of a message deletion
3766 attack.

3767 The solution to a message deletion attack is to use message sequence integrity safeguards between the
3768 actors.

3769 9.1.5 Message modification

3770 If an adversary can intercept a message and change its contents, then they may be able to alter an
3771 **authorization decision**. A message integrity safeguard can prevent a successful message modification
3772 attack.

3773 9.1.6 NotApplicable results

3774 A result of "NotApplicable" means that the **PDP** could not locate a **policy** whose **target** matched the
3775 information in the **decision request**. In general, it is highly recommended that a "Deny" **effect policy** be
3776 used, so that when a **PDP** would have returned "NotApplicable", a result of "Deny" is returned instead.

3777 In some security models, however, such as those found in many web servers, an **authorization decision**
3778 of "NotApplicable" is treated as equivalent to "Permit". There are particular security considerations that
3779 must be taken into account for this to be safe. These are explained in the following paragraphs.

3780 If "NotApplicable" is to be treated as "Permit", it is vital that the matching algorithms used by the **policy** to
3781 match elements in the **decision request** be closely aligned with the data syntax used by the applications
3782 that will be submitting the **decision request**. A failure to match will result in "NotApplicable" and be
3783 treated as "Permit". So an unintended failure to match may allow unintended **access**.

3784 Commercial http responders allow a variety of syntaxes to be treated equivalently. The "%" can be used
3785 to represent characters by hex value. The URL path "/./" provides multiple ways of specifying the same
3786 value. Multiple character sets may be permitted and, in some cases, the same printed character can be
3787 represented by different binary values. Unless the matching algorithm used by the **policy** is sophisticated
3788 enough to catch these variations, unintended **access** may be permitted.

3789 It may be safe to treat "NotApplicable" as "Permit" only in a closed environment where all applications that
3790 formulate a **decision request** can be guaranteed to use the exact syntax expected by the **policies**. In a
3791 more open environment, where **decision requests** may be received from applications that use any legal
3792 syntax, it is strongly recommended that "NotApplicable" NOT be treated as "Permit" unless matching
3793 **rules** have been very carefully designed to match all possible applicable inputs, regardless of syntax or
3794 type variations. Note, however, that according to Section 7.2, a **PEP** must deny **access** unless it
3795 receives an explicit "Permit" **authorization decision**.

3796 9.1.7 Negative rules

3797 A negative **rule** is one that is based on a **predicate** not being "True". If not used with care, negative
3798 **rules** can lead to policy violations, therefore some authorities recommend that they not be used.
3799 However, negative **rules** can be extremely efficient in certain cases, so XACML has chosen to include
3800 them. Nevertheless, it is recommended that they be used with care and avoided if possible.

3801 A common use for negative **rules** is to deny **access** to an individual or subgroup when their membership
3802 in a larger group would otherwise permit them **access**. For example, we might want to write a **rule** that
3803 allows all vice presidents to see the unpublished financial data, except for Joe, who is only a ceremonial
3804 vice president and can be indiscreet in his communications. If we have complete control over the
3805 administration of **subject attributes**, a superior approach would be to define "Vice President" and
3806 "Ceremonial Vice President" as distinct groups and then define **rules** accordingly. However, in some

3807 environments this approach may not be feasible. (It is worth noting in passing that referring to individuals
3808 in **rules** does not scale well. Generally, shared **attributes** are preferred.)

3809 If not used with care, negative **rules** can lead to policy violations in two common cases: when **attributes**
3810 are suppressed and when the base group changes. An example of suppressed **attributes** would be if we
3811 have a **policy** that **access** should be permitted, unless the **subject** is a credit risk. If it is possible that
3812 the **attribute** of being a credit risk may be unknown to the **PDP** for some reason, then unauthorized
3813 **access** may result. In some environments, the **subject** may be able to suppress the publication of
3814 **attributes** by the application of privacy controls, or the server or repository that contains the information
3815 may be unavailable for accidental or intentional reasons.

3816 An example of a changing base group would be if there is a **policy** that everyone in the engineering
3817 department may change software source code, except for secretaries. Suppose now that the department
3818 was to merge with another engineering department and the intent is to maintain the same **policy**.
3819 However, the new department also includes individuals identified as administrative assistants, who ought
3820 to be treated in the same way as secretaries. Unless the **policy** is altered, they will unintentionally be
3821 permitted to change software source code. Problems of this type are easy to avoid when one individual
3822 administers all **policies**, but when administration is distributed, as XACML allows, this type of situation
3823 must be explicitly guarded against.

3824 9.1.8 Denial of service

3825 A denial of service attack is one in which the adversary overloads an XACML actor with excessive
3826 computations or network traffic such that legitimate users cannot access the services provided by the
3827 actor.

3828 The urn:oasis:names:tc:xacml:3.0:function:access-permitted function may lead to hard to predict behavior
3829 in the **PDP**. It is possible that the function is invoked during the recursive invocations of the **PDP** such that
3830 loops are formed. Such loops may in some cases lead to large numbers of requests to be generated
3831 before the **PDP** can detect the loop and abort evaluation. Such loops could cause a denial of service at
3832 the **PDP**, either because of a malicious **policy** or because of a mistake in a **policy**.

3833 9.2 Safeguards

3834 9.2.1 Authentication

3835 Authentication provides the means for one party in a transaction to determine the identity of the other
3836 party in the transaction. Authentication may be in one direction, or it may be bilateral.

3837 Given the sensitive nature of **access control** systems, it is important for a **PEP** to authenticate the
3838 identity of the **PDP** to which it sends **decision requests**. Otherwise, there is a risk that an adversary
3839 could provide false or invalid **authorization decisions**, leading to a policy violation.

3840 It is equally important for a **PDP** to authenticate the identity of the **PEP** and assess the level of trust to
3841 determine what, if any, sensitive data should be passed. One should keep in mind that even simple
3842 "Permit" or "Deny" responses could be exploited if an adversary were allowed to make unlimited requests
3843 to a **PDP**.

3844 Many different techniques may be used to provide authentication, such as co-located code, a private
3845 network, a VPN, or digital signatures. Authentication may also be performed as part of the
3846 communication protocol used to exchange the **contexts**. In this case, authentication may be performed
3847 either at the message level or at the session level.

3848 9.2.2 Policy administration

3849 If the contents of **policies** are exposed outside of the **access control** system, potential **subjects** may
3850 use this information to determine how to gain unauthorized **access**.

3851 To prevent this threat, the repository used for the storage of **policies** may itself require **access control**.
3852 In addition, the <Status> element should be used to return values of missing **attributes** only when
3853 exposure of the identities of those **attributes** will not compromise security.

3854 9.2.3 Confidentiality

3855 Confidentiality mechanisms ensure that the contents of a message can be read only by the desired
3856 recipients and not by anyone else who encounters the message while it is in transit. There are two areas
3857 in which confidentiality should be considered: one is confidentiality during transmission; the other is
3858 confidentiality within a <Policy> element.

3859 9.2.3.1 Communication confidentiality

3860 In some environments it is deemed good practice to treat all data within an **access control** system as
3861 confidential. In other environments, **policies** may be made freely available for distribution, inspection,
3862 and audit. The idea behind keeping **policy** information secret is to make it more difficult for an adversary
3863 to know what steps might be sufficient to obtain unauthorized **access**. Regardless of the approach
3864 chosen, the security of the **access control** system should not depend on the secrecy of the **policy**.

3865 Any security considerations related to transmitting or exchanging XACML <Policy> elements are
3866 outside the scope of the XACML standard. While it is important to ensure that the integrity and
3867 confidentiality of <Policy> elements is maintained when they are exchanged between two parties, it is
3868 left to the implementers to determine the appropriate mechanisms for their environment.

3869 Communications confidentiality can be provided by a confidentiality mechanism, such as SSL. Using a
3870 point-to-point scheme like SSL may lead to other vulnerabilities when one of the end-points is
3871 compromised.

3872 9.2.3.2 Statement level confidentiality

3873 In some cases, an implementation may want to encrypt only parts of an XACML <Policy> element.

3874 The XML Encryption Syntax and Processing Candidate Recommendation from W3C can be used to
3875 encrypt all or parts of an XML document. This specification is recommended for use with XACML.

3876 It should go without saying that if a repository is used to facilitate the communication of cleartext (i.e.,
3877 unencrypted) **policy** between the **PAP** and **PDP**, then a secure repository should be used to store this
3878 sensitive data.

3879 9.2.4 Policy integrity

3880 The XACML **policy** used by the **PDP** to evaluate the request **context** is the heart of the system.
3881 Therefore, maintaining its integrity is essential. There are two aspects to maintaining the integrity of the
3882 **policy**. One is to ensure that <Policy> elements have not been altered since they were originally
3883 created by the **PAP**. The other is to ensure that <Policy> elements have not been inserted or deleted
3884 from the set of **policies**.

3885 In many cases, both aspects can be achieved by ensuring the integrity of the actors and implementing
3886 session-level mechanisms to secure the communication between actors. The selection of the appropriate
3887 mechanisms is left to the implementers. However, when **policy** is distributed between organizations to
3888 be acted on at a later time, or when the **policy** travels with the protected **resource**, it would be useful to
3889 sign the **policy**. In these cases, the XML Signature Syntax and Processing standard from W3C is
3890 recommended to be used with XACML.

3891 Digital signatures should only be used to ensure the integrity of the statements. Digital signatures should
3892 not be used as a method of selecting or evaluating **policy**. That is, the **PDP** should not request a **policy**
3893 based on who signed it or whether or not it has been signed (as such a basis for selection would, itself,
3894 be a matter of policy). However, the **PDP** must verify that the key used to sign the **policy** is one
3895 controlled by the purported **issuer** of the **policy**. The means to do this are dependent on the specific
3896 signature technology chosen and are outside the scope of this document.

3897 9.2.5 Policy identifiers

3898 Since **policies** can be referenced by their identifiers, it is the responsibility of the **PAP** to ensure that
3899 these are unique. Confusion between identifiers could lead to misidentification of the **applicable policy**.

3900 This specification is silent on whether a **PAP** must generate a new identifier when a **policy** is modified or
3901 may use the same identifier in the modified **policy**. This is a matter of administrative practice. However,
3902 care must be taken in either case. If the identifier is reused, there is a danger that other **policies** or
3903 **policy sets** that reference it may be adversely affected. Conversely, if a new identifier is used, these
3904 other **policies** may continue to use the prior **policy**, unless it is deleted. In either case the results may
3905 not be what the **policy** administrator intends.

3906 If a **PDP** is provided with **policies** from distinct sources which might not be fully trusted, as in the use of
3907 the administration profile [**XACMLAdmin**], there is a concern that someone could intentionally publish a
3908 **policy** with an id which collides with another **policy**. This could cause **policy** references that point to the
3909 wrong **policy**, and may cause other unintended consequences in an implementation which is predicated
3910 upon having unique **policy** identifiers.

3911 If this issue is a concern it is RECOMMENDED that distinct **policy** issuers or sources are assigned
3912 distinct namespaces for **policy** identifiers. One method is to make sure that the **policy** identifier begins
3913 with a string which has been assigned to the particular **policy** issuer or source. The remainder of the
3914 **policy** identifier is an issuer-specific unique part. For instance, Alice from Example Inc. could be assigned
3915 the **policy** identifiers which begin with `http://example.com/xacml/policyId/alice/`. The **PDP** or another
3916 trusted component can then verify that the authenticated source of the **policy** is Alice at Example Inc, or
3917 otherwise reject the **policy**. Anyone else will be unable to publish **policies** with identifiers which collide
3918 with the **policies** of Alice.

3919 9.2.6 Trust model

3920 Discussions of authentication, integrity and confidentiality safeguards necessarily assume an underlying
3921 trust model: how can one actor come to believe that a given key is uniquely associated with a specific,
3922 identified actor so that the key can be used to encrypt data for that actor or verify signatures (or other
3923 integrity structures) from that actor? Many different types of trust models exist, including strict
3924 hierarchies, distributed authorities, the Web, the bridge, and so on.

3925 It is worth considering the relationships between the various actors of the **access control** system in terms
3926 of the interdependencies that do and do not exist.

- 3927 • None of the entities of the authorization system are dependent on the **PEP**. They may collect data
3928 from it, (for example authentication data) but are responsible for verifying it themselves.
- 3929 • The correct operation of the system depends on the ability of the **PEP** to actually enforce **policy**
3930 **decisions**.
- 3931 • The **PEP** depends on the **PDP** to correctly evaluate **policies**. This in turn implies that the **PDP** is
3932 supplied with the correct inputs. Other than that, the **PDP** does not depend on the **PEP**.
- 3933 • The **PDP** depends on the **PAP** to supply appropriate **policies**. The **PAP** is not dependent on other
3934 components.

3935 9.2.7 Privacy

3936 It is important to be aware that any transactions that occur with respect to **access control** may reveal
3937 private information about the actors. For example, if an XACML **policy** states that certain data may only
3938 be read by **subjects** with "Gold Card Member" status, then any transaction in which a **subject** is
3939 permitted **access** to that data leaks information to an adversary about the **subject's** status. Privacy
3940 considerations may therefore lead to encryption and/or to **access control** requirements surrounding the
3941 enforcement of XACML **policy** instances themselves: confidentiality-protected channels for the
3942 request/response protocol messages, protection of **subject attributes** in storage and in transit, and so
3943 on.

3944 Selection and use of privacy mechanisms appropriate to a given environment are outside the scope of
3945 XACML. The **decision** regarding whether, how, and when to deploy such mechanisms is left to the
3946 implementers associated with the environment.

3947 **9.3 Unicode security issues**

3948 There are many security considerations related to use of Unicode. An XACML implementation SHOULD
3949 follow the advice given in the relevant version of **[UTR36]**.

3950 **9.4 Identifier equality**

3951 Section 7.20 defines the identifier equality operation for XACML. This definition of equality does not do
3952 any kind of canonicalization or escaping of characters. The identifiers defined in the XACML specification
3953 have been selected to not include any ambiguity regarding these aspects. It is RECOMMENDED that
3954 identifiers defined by extensions also do not introduce any identifiers which might be mistaken for being
3955 subject to processing, like for instance URL character encoding using “%”.

3956 **10 Conformance**

3957 **10.1 Introduction**

3958 The XACML specification addresses the following aspect of conformance:

3959 The XACML specification defines a number of functions, etc. that have somewhat special applications,
3960 therefore they are not required to be implemented in an implementation that claims to conform with the
3961 OASIS standard.

3962 **10.2 Conformance tables**

3963 This section lists those portions of the specification that **MUST** be included in an implementation of a **PDP**
3964 that claims to conform to XACML v3.0. A set of test cases has been created to assist in this process.
3965 These test cases can be located from the OASIS XACML TC Web page. The site hosting the test cases
3966 contains a full description of the test cases and how to execute them.

3967 Note: "M" means mandatory-to-implement. "O" means optional.

3968 The implementation **MUST** follow sections 5, 6, 7, Appendix A, Appendix B and Appendix C where they
3969 apply to implemented items in the following tables.

3970 **10.2.1 Schema elements**

3971 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

3972 **10.2.2 Identifier Prefixes**

3973 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

3974 **10.2.3 Algorithms**

3975 The implementation MUST include the **rule-** and **policy-combining algorithms** associated with the
3976 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

3977 10.2.4 Status Codes

3978 Implementation support for the <StatusCode> element is optional, but if the element is supported, then
3979 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

3980 10.2.5 Attributes

3981 The implementation MUST support the **attributes** associated with the following identifiers as specified by
3982 XACML. If values for these **attributes** are not present in the **decision request**, then their values MUST
3983 be supplied by the **context handler**. So, unlike most other **attributes**, their semantics are not
3984 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

3985 10.2.6 Identifiers

3986 The implementation MUST use the **attributes** associated with the following identifiers in the way XACML
3987 has defined. This requirement pertains primarily to implementations of a **PAP** or **PEP** that uses XACML,
3988 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
<u>urn:oasis:names:tc:xacml:2.0:resource:target-namespace</u>	O
urn:oasis:names:tc:xacml:1.0:action:action-id	
<u>urn:oasis:names:tc:xacml:1.0:action:action-namespace</u>	O
urn:oasis:names:tc:xacml:1.0:action:implied-action	

3989 10.2.7 Data-types

3990 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

3991 10.2.8 Functions

3992 The implementation MUST properly process those functions associated with the identifiers marked with
3993 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

3994 **10.2.9 Identifiers planned for future deprecation**

3995 These identifiers are associated with previous versions of XACML and newer alternatives exist in XACML
3996 3.0. They are planned to be deprecated at some unspecified point in the future. It is RECOMMENDED
3997 that these identifiers not be used in new policies and requests.

3998 The implementation MUST properly process those features associated with the identifiers marked with an
3999 "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

4000

4001 Appendix A. Data-types and functions (normative)

4002 A.1 Introduction

4003 This section specifies the data-types and functions used in XACML to create *predicates* for *conditions*
4004 and *target* matches.

4005 This specification combines the various standards set forth by IEEE and ANSI for string representation of
4006 numeric values, as well as the evaluation of arithmetic functions. It describes the primitive data-types and
4007 *bags*. The standard functions are named and their operational semantics are described.

4008 A.2 Data-types

4009 Although XML instances represent all data-types as strings, an XACML *PDP* must operate on types of
4010 data that, while they have string representations, are not just strings. Types such as Boolean, integer,
4011 and double MUST be converted from their XML string representations to values that can be compared
4012 with values in their domain of discourse, such as numbers. The following primitive data-types are
4013 specified for use with XACML and have explicit data representations:

- 4014 • <http://www.w3.org/2001/XMLSchema#string>
- 4015 • <http://www.w3.org/2001/XMLSchema#boolean>
- 4016 • <http://www.w3.org/2001/XMLSchema#integer>
- 4017 • <http://www.w3.org/2001/XMLSchema#double>
- 4018 • <http://www.w3.org/2001/XMLSchema#time>
- 4019 • <http://www.w3.org/2001/XMLSchema#date>
- 4020 • <http://www.w3.org/2001/XMLSchema#dateTime>
- 4021 • <http://www.w3.org/2001/XMLSchema#anyURI>
- 4022 • <http://www.w3.org/2001/XMLSchema#hexBinary>
- 4023 • <http://www.w3.org/2001/XMLSchema#base64Binary>
- 4024 • <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- 4025 • <http://www.w3.org/2001/XMLSchema#yearMonthDuration>
- 4026 • <urn:oasis:names:tc:xacml:1.0:data-type:x500Name>
- 4027 • <urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name>
- 4028 • <urn:oasis:names:tc:xacml:2.0:data-type:ipAddress>
- 4029 • <urn:oasis:names:tc:xacml:2.0:data-type:dnsName>
- 4030 • <urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression>

4031 For the sake of improved interoperability, it is RECOMMENDED that all time references be in UTC time.

4032 An XACML *PDP* SHALL be capable of converting string representations into various primitive data-types.
4033 For doubles, XACML SHALL use the conversions described in [IEEE754].

4034 XACML defines four data-types representing identifiers for *subjects* or *resources*; these are:

- 4035 “urn:oasis:names:tc:xacml:1.0:data-type:x500Name”,
- 4036 “urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name”
- 4037 “urn:oasis:names:tc:xacml:2.0:data-type:ipAddress” and
- 4038 “urn:oasis:names:tc:xacml:2.0:data-type:dnsName”

4039 These types appear in several standard applications, such as TLS/SSL and electronic mail.

4040 X.500 directory name

4041 The "urn:oasis:names:tc:xacml:1.0:data-type:x500Name" primitive type represents an ITU-T Rec.
4042 X.520 Distinguished Name. The valid syntax for such a name is described in IETF RFC 2253
4043 "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished
4044 Names".

4045 **RFC 822 name**

4046 The "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" primitive type represents an electronic
4047 mail address. The valid syntax for such a name is described in IETF RFC 2821, Section 4.1.2,
4048 Command Argument Syntax, under the term "Mailbox".

4049 **IP address**

4050 The "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" primitive type represents an IPv4 or IPv6
4051 network address, with optional mask and optional port or port range. The syntax SHALL be:

4052 ipAddress = address ["/" mask] [":" [portrange]]

4053 For an IPv4 address, the address and mask are formatted in accordance with the syntax for a
4054 "host" in IETF RFC 2396 "Uniform Resource Identifiers (URI): Generic Syntax", section 3.2.

4055 For an IPv6 address, the address and mask are formatted in accordance with the syntax for an
4056 "ipv6reference" in IETF RFC 2732 "Format for Literal IPv6 Addresses in URL's". (Note that an
4057 IPv6 address or mask, in this syntax, is enclosed in literal "[" "]" brackets.)

4058 **DNS name**

4059 The "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" primitive type represents a Domain
4060 Name Service (DNS) host name, with optional port or port range. The syntax SHALL be:

4061 dnsName = hostname [":" portrange]

4062 The hostname is formatted in accordance with IETF RFC 2396 "Uniform Resource Identifiers
4063 (URI): Generic Syntax", section 3.2, except that a wildcard "*" may be used in the left-most
4064 component of the hostname to indicate "any subdomain" under the domain specified to its right.

4065 For both the "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" and
4066 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" data-types, the port or port range syntax
4067 SHALL be

4068 portrange = portnumber | "-"portnumber | portnumber "-"portnumber

4069 where "portnumber" is a decimal port number. If the port number is of the form "-x", where "x" is
4070 a port number, then the range is all ports numbered "x" and below. If the port number is of the
4071 form "x-", then the range is all ports numbered "x" and above. [This syntax is taken from the Java
4072 SocketPermission.]

4073 **XPath expression**

4074 The "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" primitive type represents an
4075 XPath expression over the XML in a <Content> element. The syntax is defined by the XPath
4076 W3C recommendation. The content of this data type also includes the context in which
4077 namespaces prefixes in the expression are resolved, which distinguishes it from a plain string and
4078 the XACML **attribute** category of the <Content> element to which it applies. When the value is
4079 encoded in an <AttributeValue> element, the namespace context is given by the [in-scope](#)
4080 [namespaces](#) (see [INFOSET](#)) of the <AttributeValue> element, and an XML attribute called
4081 XPathCategory gives the category of the <Content> element where the expression applies.

4082 The XPath expression MUST be evaluated in a context which is equivalent of a stand alone XML
4083 document with the only child of the <Content> element as the document element. Namespace
4084 declarations which are not "visibly utilized", as defined by [\[exc-c14n\]](#), MAY not be present and
4085 MUST NOT be utilized by the XPath expression. The context node of the XPath expression is the
4086 document node of this stand alone document.

4087 A.3 Functions

4088 XACML specifies the following functions. Unless otherwise specified, if an argument of one of these
4089 functions were to evaluate to "Indeterminate", then the function SHALL be set to "Indeterminate".

4090 Note that in each case an implementation is conformant as long as it produces the same result as is
4091 specified here, regardless of how and in what order the implementation behaves internally.

4092 A.3.1 Equality predicates

4093 The following functions are the equality functions for the various primitive types. Each function for a
4094 particular data-type follows a specified standard convention for that data-type.

- 4095 • urn:oasis:names:tc:xacml:1.0:function:string-equal

4096 This function SHALL take two arguments of data-type
4097 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
4098 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if
4099 the value of both of its arguments are of equal length and each string is determined to be equal.
4100 Otherwise, it SHALL return "False". The comparison SHALL use Unicode codepoint collation, as
4101 defined for the identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.

- 4102 • urn:oasis:names:tc:xacml:3.0:function:string-equal-ignore-case

4103 This function SHALL take two arguments of data-type
4104 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
4105 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be "True" if and only if the
4106 two strings are equal as defined by urn:oasis:names:tc:xacml:1.0:function:string-equal after they
4107 have both been converted to lower case with urn:oasis:names:tc:xacml:1.0:function:string-
4108 normalize-to-lower-case.

- 4109 • urn:oasis:names:tc:xacml:1.0:function:boolean-equal

4110 This function SHALL take two arguments of data-type
4111 "http://www.w3.org/2001/XMLSchema#boolean" and SHALL return an
4112 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if
4113 the arguments are equal. Otherwise, it SHALL return "False".

- 4114 • urn:oasis:names:tc:xacml:1.0:function:integer-equal

4115 This function SHALL take two arguments of data-type
4116 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return an
4117 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and only if
4118 the two arguments represent the same number.

- 4119 • urn:oasis:names:tc:xacml:1.0:function:double-equal

4120 This function SHALL take two arguments of data-type
4121 "http://www.w3.org/2001/XMLSchema#double" and SHALL return an
4122 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation on doubles
4123 according to IEEE 754 **[IEEE754]**.

- 4124 • urn:oasis:names:tc:xacml:1.0:function:date-equal

4125 This function SHALL take two arguments of data-type
4126 "http://www.w3.org/2001/XMLSchema#date" and SHALL return an
4127 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to
4128 the "op:date-equal" function **[XF]** Section 10.4.9.

- 4129 • urn:oasis:names:tc:xacml:1.0:function:time-equal

4130 This function SHALL take two arguments of data-type
4131 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
4132 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to
4133 the "op:time-equal" function **[XF]** Section 10.4.12.

- 4134 • urn:oasis:names:tc:xacml:1.0:function:dateTime-equal
4135 This function SHALL take two arguments of data-type
4136 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an
4137 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation according to
4138 the "op:dateTime-equal" function **[XF]** Section 10.4.6.
- 4139 • urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-equal
4140 This function SHALL take two arguments of data-type
4141 "http://www.w3.org/2001/XMLSchema#dayTimeDuration" and SHALL return an
4142 "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation
4143 according to the "op:duration-equal" function **[XF]** Section 10.4.5. Note that the lexical
4144 representation of each argument MUST be converted to a value expressed in fractional seconds
4145 **[XF]** Section 10.3.2.
- 4146 • urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-equal
4147 This function SHALL take two arguments of data-type
4148 "http://www.w3.org/2001/XMLSchema#yearMonthDuration" and SHALL return an
4149 "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation
4150 according to the "op:duration-equal" function **[XF]** Section 10.4.5. Note that the lexical
4151 representation of each argument MUST be converted to a value expressed in fractional seconds
4152 **[XF]** Section 10.3.2.
- 4153 • urn:oasis:names:tc:xacml:1.0:function:anyURI-equal
4154 This function SHALL take two arguments of data-type
4155 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return an
4156 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL convert the arguments to
4157 strings with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI and return "True" if and
4158 only if the values of the two arguments are equal on a codepoint-by-codepoint basis.
- 4159 • urn:oasis:names:tc:xacml:1.0:function:x500Name-equal
4160 This function SHALL take two arguments of "urn:oasis:names:tc:xacml:1.0:data-type:x500Name"
4161 and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if
4162 and only if each Relative Distinguished Name (RDN) in the two arguments matches. Otherwise,
4163 it SHALL return "False". Two RDNs shall be said to match if and only if the result of the following
4164 operations is "True" .
- 4165 1. Normalize the two arguments according to IETF RFC 2253 "Lightweight Directory Access
4166 Protocol (v3): UTF-8 String Representation of Distinguished Names".
 - 4167 2. If any RDN contains multiple attributeTypeAndValue pairs, re-order the Attribute
4168 ValuePairs in that RDN in ascending order when compared as octet strings (described in
4169 ITU-T Rec. X.690 (1997 E) Section 11.6 "Set-of components").
 - 4170 3. Compare RDNs using the rules in IETF RFC 3280 "Internet X.509 Public Key
4171 Infrastructure Certificate and Certificate Revocation List (CRL) Profile", Section 4.1.2.4
4172 "Issuer".
- 4173 • urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal
4174 This function SHALL take two arguments of data-type "urn:oasis:names:tc:xacml:1.0:data-
4175 type:rfc822Name" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It
4176 SHALL return "True" if and only if the two arguments are equal. Otherwise, it SHALL return
4177 "False". An RFC822 name consists of a local-part followed by "@" followed by a domain-part.
4178 The local-part is case-sensitive, while the domain-part (which is usually a DNS host name) is not
4179 case-sensitive. Perform the following operations:
- 4180 1. Normalize the domain-part of each argument to lower case
 - 4181 2. Compare the expressions by applying the function
4182 "urn:oasis:names:tc:xacml:1.0:function:string-equal" to the normalized arguments.

4183 • urn:oasis:names:tc:xacml:1.0:function:hexBinary-equal
4184 This function SHALL take two arguments of data-type
4185 "http://www.w3.org/2001/XMLSchema#hexBinary" and SHALL return an
4186 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet sequences
4187 represented by the value of both arguments have equal length and are equal in a conjunctive,
4188 point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function.
4189 Otherwise, it SHALL return "False". The conversion from the string representation to an octet
4190 sequence SHALL be as specified in [XS] Section 3.2.15.

4191 • urn:oasis:names:tc:xacml:1.0:function:base64Binary-equal
4192 This function SHALL take two arguments of data-type
4193 "http://www.w3.org/2001/XMLSchema#base64Binary" and SHALL return an
4194 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet sequences
4195 represented by the value of both arguments have equal length and are equal in a conjunctive,
4196 point-wise, comparison using the "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function.
4197 Otherwise, it SHALL return "False". The conversion from the string representation to an octet
4198 sequence SHALL be as specified in [XS] Section 3.2.16.

4199 **A.3.2 Arithmetic functions**

4200 All of the following functions SHALL take two arguments of the specified data-type, integer, or double,
4201 and SHALL return an element of integer or double data-type, respectively. However, the "add" and
4202 "multiply" functions MAY take more than two arguments. Each function evaluation operating on doubles
4203 SHALL proceed as specified by their logical counterparts in IEEE 754 [IEEE754]. For all of these
4204 functions, if any argument is "Indeterminate", then the function SHALL evaluate to "Indeterminate". In the
4205 case of the divide functions, if the divisor is zero, then the function SHALL evaluate to "Indeterminate".

4206 • urn:oasis:names:tc:xacml:1.0:function:integer-add
4207 This function MUST accept two or more arguments.

4208 • urn:oasis:names:tc:xacml:1.0:function:double-add
4209 This function MUST accept two or more arguments.

4210 • urn:oasis:names:tc:xacml:1.0:function:integer-subtract
4211 The result is the second argument subtracted from the first argument.

4212 • urn:oasis:names:tc:xacml:1.0:function:double-subtract
4213 The result is the second argument subtracted from the first argument.

4214 • urn:oasis:names:tc:xacml:1.0:function:integer-multiply
4215 This function MUST accept two or more arguments.

4216 • urn:oasis:names:tc:xacml:1.0:function:double-multiply
4217 This function MUST accept two or more arguments.

4218 • urn:oasis:names:tc:xacml:1.0:function:integer-divide
4219 The result is the first argument divided by the second argument.

4220 • urn:oasis:names:tc:xacml:1.0:function:double-divide
4221 The result is the first argument divided by the second argument.

4222 • urn:oasis:names:tc:xacml:1.0:function:integer-mod
4223 The result is remainder of the first argument divided by the second argument.

4224 The following functions SHALL take a single argument of the specified data-type. The round and floor
4225 functions SHALL take a single argument of data-type "http://www.w3.org/2001/XMLSchema#double" and
4226 return a value of the data-type "http://www.w3.org/2001/XMLSchema#double".

4227 • urn:oasis:names:tc:xacml:1.0:function:integer-abs

- 4228 • urn:oasis:names:tc:xacml:1.0:function:double-abs
- 4229 • urn:oasis:names:tc:xacml:1.0:function:round
- 4230 • urn:oasis:names:tc:xacml:1.0:function:floor

4231 **A.3.3 String conversion functions**

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

- 4234 • urn:oasis:names:tc:xacml:1.0:function:string-normalize-space
 - 4235 This function SHALL take one argument of data-type
 - 4236 “http://www.w3.org/2001/XMLSchema#string” and SHALL normalize the value by stripping off all
 - 4237 leading and trailing white space characters. The whitespace characters are defined in the
 - 4238 metasympol S (Production 3) of [XML].
- 4239 • urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case
 - 4240 This function SHALL take one argument of data-type
 - 4241 “http://www.w3.org/2001/XMLSchema#string” and SHALL normalize the value by converting each
 - 4242 upper case character to its lower case equivalent. Case mapping shall be done as specified for
 - 4243 the fn:lower-case function in [XF] with no tailoring for particular languages or environments.

4244 **A.3.4 Numeric data-type conversion functions**

4245 The following functions convert between the data-type “http://www.w3.org/2001/XMLSchema#integer”
4246 and” http://www.w3.org/2001/XMLSchema#double” primitive types.

- 4247 • urn:oasis:names:tc:xacml:1.0:function:double-to-integer
 - 4248 This function SHALL take one argument of data-type
 - 4249 “http://www.w3.org/2001/XMLSchema#double” and SHALL truncate its numeric value to a whole
 - 4250 number and return an element of data-type “http://www.w3.org/2001/XMLSchema#integer”.
- 4251 • urn:oasis:names:tc:xacml:1.0:function:integer-to-double
 - 4252 This function SHALL take one argument of data-type
 - 4253 “http://www.w3.org/2001/XMLSchema#integer” and SHALL promote its value to an element of
 - 4254 data-type “http://www.w3.org/2001/XMLSchema#double” with the same numeric value. If the
 - 4255 integer argument is outside the range which can be represented by a double, the result SHALL
 - 4256 be Indeterminate, with the status code “urn:oasis:names:tc:xacml:1.0:status:processing-error”.

4257 **A.3.5 Logical functions**

4258 This section contains the specification for logical functions that operate on arguments of data-type
4259 “http://www.w3.org/2001/XMLSchema#boolean”.

- 4260 • urn:oasis:names:tc:xacml:1.0:function:or
 - 4261 This function SHALL return "False" if it has no arguments and SHALL return "True" if at least one
 - 4262 of its arguments evaluates to "True". The order of evaluation SHALL be from first argument to
 - 4263 last. The evaluation SHALL stop with a result of "True" if any argument evaluates to "True",
 - 4264 leaving the rest of the arguments unevaluated.
- 4265 • urn:oasis:names:tc:xacml:1.0:function:and
 - 4266 This function SHALL return "True" if it has no arguments and SHALL return "False" if one of its
 - 4267 arguments evaluates to "False". The order of evaluation SHALL be from first argument to last.
 - 4268 The evaluation SHALL stop with a result of "False" if any argument evaluates to "False", leaving
 - 4269 the rest of the arguments unevaluated.
- 4270 • urn:oasis:names:tc:xacml:1.0:function:n-of
 - 4271 The first argument to this function SHALL be of data-type
 - 4272 http://www.w3.org/2001/XMLSchema#integer. The remaining arguments SHALL be of data-type

4273 http://www.w3.org/2001/XMLSchema#boolean. The first argument specifies the minimum
4274 number of the remaining arguments that MUST evaluate to "True" for the expression to be
4275 considered "True". If the first argument is 0, the result SHALL be "True". If the number of
4276 arguments after the first one is less than the value of the first argument, then the expression
4277 SHALL result in "Indeterminate". The order of evaluation SHALL be: first evaluate the integer
4278 value, and then evaluate each subsequent argument. The evaluation SHALL stop and return
4279 "True" if the specified number of arguments evaluate to "True". The evaluation of arguments
4280 SHALL stop if it is determined that evaluating the remaining arguments will not satisfy the
4281 requirement.

4282 • urn:oasis:names:tc:xacml:1.0:function:not

4283 This function SHALL take one argument of data-type
4284 "http://www.w3.org/2001/XMLSchema#boolean". If the argument evaluates to "True", then the
4285 result of the expression SHALL be "False". If the argument evaluates to "False", then the result
4286 of the expression SHALL be "True".

4287 Note: When evaluating and, or, or n-of, it **MAY NOT** **may not** be necessary to attempt a full evaluation of
4288 each argument in order to determine whether the evaluation of the argument would result in
4289 "Indeterminate". Analysis of the argument regarding the availability of its **attributes**, or other analysis
4290 regarding errors, such as "divide-by-zero", may render the argument error free. Such arguments
4291 occurring in the expression in a position after the evaluation is stated to stop need not be processed.

4292 **A.3.6 Numeric comparison functions**

4293 These functions form a minimal set for comparing two numbers, yielding a Boolean result. For doubles
4294 they SHALL comply with the rules governed by IEEE 754 [IEEE754].

- 4295 • urn:oasis:names:tc:xacml:1.0:function:integer-greater-than
- 4296 • urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal
- 4297 • urn:oasis:names:tc:xacml:1.0:function:integer-less-than
- 4298 • urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal
- 4299 • urn:oasis:names:tc:xacml:1.0:function:double-greater-than
- 4300 • urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal
- 4301 • urn:oasis:names:tc:xacml:1.0:function:double-less-than
- 4302 • urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal

4303 **A.3.7 Date and time arithmetic functions**

4304 These functions perform arithmetic operations with date and time.

4305 • urn:oasis:names:tc:xacml:3.0:function:dateTime-add-dayTimeDuration

4306 This function SHALL take two arguments, the first SHALL be of data-type
4307 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be of data-type
4308 "http://www.w3.org/2001/XMLSchema#dayTimeDuration". It SHALL return a result of
4309 "http://www.w3.org/2001/XMLSchema#dateTime". This function SHALL return the value by
4310 adding the second argument to the first argument according to the specification of adding
4311 durations to date and time [XS] Appendix E.

4312 • urn:oasis:names:tc:xacml:3.0:function:dateTime-add-yearMonthDuration

4313 This function SHALL take two arguments, the first SHALL be a
4314 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a
4315 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
4316 "http://www.w3.org/2001/XMLSchema#dateTime". This function SHALL return the value by
4317 adding the second argument to the first argument according to the specification of adding
4318 durations to date and time [XS] Appendix E.

- 4319 • urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-dayTimeDuration
 - 4320 This function SHALL take two arguments, the first SHALL be a
 - 4321 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a
 - 4322 "http://www.w3.org/2001/XMLSchema#dayTimeDuration". It SHALL return a result of
 - 4323 "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration,
 - 4324 then this function SHALL return the value by adding the corresponding negative duration, as per
 - 4325 the specification [XS] Appendix E. If the second argument is a negative duration, then the result
 - 4326 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-
 - 4327 dayTimeDuration" had been applied to the corresponding positive duration.
- 4328 • urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-yearMonthDuration
 - 4329 This function SHALL take two arguments, the first SHALL be a
 - 4330 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a
 - 4331 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
 - 4332 "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument is a positive duration,
 - 4333 then this function SHALL return the value by adding the corresponding negative duration, as per
 - 4334 the specification [XS] Appendix E. If the second argument is a negative duration, then the result
 - 4335 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:dateTime-add-
 - 4336 yearMonthDuration" had been applied to the corresponding positive duration.
- 4337 • urn:oasis:names:tc:xacml:3.0:function:date-add-yearMonthDuration
 - 4338 This function SHALL take two arguments, the first SHALL be a
 - 4339 "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a
 - 4340 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
 - 4341 "http://www.w3.org/2001/XMLSchema#date". This function SHALL return the value by adding the
 - 4342 second argument to the first argument according to the specification of adding duration to date
 - 4343 [XS] Appendix E.
- 4344 • urn:oasis:names:tc:xacml:3.0:function:date-subtract-yearMonthDuration
 - 4345 This function SHALL take two arguments, the first SHALL be a
 - 4346 "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a
 - 4347 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". It SHALL return a result of
 - 4348 "http://www.w3.org/2001/XMLSchema#date". If the second argument is a positive duration, then
 - 4349 this function SHALL return the value by adding the corresponding negative duration, as per the
 - 4350 specification [XS] Appendix E. If the second argument is a negative duration, then the result
 - 4351 SHALL be as if the function "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration"
 - 4352 had been applied to the corresponding positive duration.

4353 A.3.8 Non-numeric comparison functions

4354 These functions perform comparison operations on two arguments of non-numerical types.

- 4355 • urn:oasis:names:tc:xacml:1.0:function:string-greater-than
 - 4356 This function SHALL take two arguments of data-type
 - 4357 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
 - 4358 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
 - 4359 argument is lexicographically strictly greater than the second argument. Otherwise, it SHALL
 - 4360 return "False". The comparison SHALL use Unicode codepoint collation, as defined for the
 - 4361 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [XF].
- 4362 • urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal
 - 4363 This function SHALL take two arguments of data-type
 - 4364 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
 - 4365 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
 - 4366 argument is lexicographically greater than or equal to the second argument. Otherwise, it SHALL
 - 4367 return "False". The comparison SHALL use Unicode codepoint collation, as defined for the
 - 4368 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by [XF].

- 4369 • urn:oasis:names:tc:xacml:1.0:function:string-less-than
 - 4370 This function SHALL take two arguments of data-type
 - 4371 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
 - 4372 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only the first
 - 4373 argument is lexicographically strictly less than the second argument. Otherwise, it SHALL return
 - 4374 "False". The comparison SHALL use Unicode codepoint collation, as defined for the identifier
 - 4375 http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.
- 4376 • urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal
 - 4377 This function SHALL take two arguments of data-type
 - 4378 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
 - 4379 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only the first
 - 4380 argument is lexicographically less than or equal to the second argument. Otherwise, it SHALL
 - 4381 return "False". The comparison SHALL use Unicode codepoint collation, as defined for the
 - 4382 identifier http://www.w3.org/2005/xpath-functions/collation/codepoint by **[XF]**.
- 4383 • urn:oasis:names:tc:xacml:1.0:function:time-greater-than
 - 4384 This function SHALL take two arguments of data-type
 - 4385 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
 - 4386 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
 - 4387 argument is greater than the second argument according to the order relation specified for
 - 4388 "http://www.w3.org/2001/XMLSchema#time" **[XS]** Section 3.2.8. Otherwise, it SHALL return
 - 4389 "False". Note: it is illegal to compare a time that includes a time-zone value with one that does
 - 4390 not. In such cases, the time-in-range function should be used.
- 4391 • urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal
 - 4392 This function SHALL take two arguments of data-type
 - 4393 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
 - 4394 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
 - 4395 argument is greater than or equal to the second argument according to the order relation
 - 4396 specified for "http://www.w3.org/2001/XMLSchema#time" **[XS]** Section 3.2.8. Otherwise, it
 - 4397 SHALL return "False". Note: it is illegal to compare a time that includes a time-zone value with
 - 4398 one that does not. In such cases, the time-in-range function should be used.
- 4399 • urn:oasis:names:tc:xacml:1.0:function:time-less-than
 - 4400 This function SHALL take two arguments of data-type
 - 4401 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
 - 4402 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
 - 4403 argument is less than the second argument according to the order relation specified for
 - 4404 "http://www.w3.org/2001/XMLSchema#time" **[XS]** Section 3.2.8. Otherwise, it SHALL return
 - 4405 "False". Note: it is illegal to compare a time that includes a time-zone value with one that does
 - 4406 not. In such cases, the time-in-range function should be used.
- 4407 • urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal
 - 4408 This function SHALL take two arguments of data-type
 - 4409 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
 - 4410 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
 - 4411 argument is less than or equal to the second argument according to the order relation specified
 - 4412 for "http://www.w3.org/2001/XMLSchema#time" **[XS]** Section 3.2.8. Otherwise, it SHALL return
 - 4413 "False". Note: it is illegal to compare a time that includes a time-zone value with one that does
 - 4414 not. In such cases, the time-in-range function should be used.
- 4415 • urn:oasis:names:tc:xacml:2.0:function:time-in-range
 - 4416 This function SHALL take three arguments of data-type
 - 4417 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an
 - 4418 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the first argument falls
 - 4419 in the range defined inclusively by the second and third arguments. Otherwise, it SHALL return

4420 “False”. Regardless of its value, the third argument SHALL be interpreted as a time that is equal
4421 to, or later than by less than twenty-four hours, the second argument. If no time zone is provided
4422 for the first argument, it SHALL use the default time zone at the **context handler**. If no time zone
4423 is provided for the second or third arguments, then they SHALL use the time zone from the first
4424 argument.

- 4425 • urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than

4426 This function SHALL take two arguments of data-type
4427 “http://www.w3.org/2001/XMLSchema#dateTime” and SHALL return an
4428 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4429 argument is greater than the second argument according to the order relation specified for
4430 “http://www.w3.org/2001/XMLSchema#dateTime” by [XS] part 2, section 3.2.7. Otherwise, it
4431 SHALL return “False”. Note: if a dateTime value does not include a time-zone value, then an
4432 implicit time-zone value SHALL be assigned, as described in [XS].

- 4433 • urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal

4434 This function SHALL take two arguments of data-type
4435 “http://www.w3.org/2001/XMLSchema#dateTime” and SHALL return an
4436 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4437 argument is greater than or equal to the second argument according to the order relation
4438 specified for “http://www.w3.org/2001/XMLSchema#dateTime” by [XS] part 2, section 3.2.7.
4439 Otherwise, it SHALL return “False”. Note: if a dateTime value does not include a time-zone
4440 value, then an implicit time-zone value SHALL be assigned, as described in [XS].

- 4441 • urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than

4442 This function SHALL take two arguments of data-type
4443 “http://www.w3.org/2001/XMLSchema#dateTime” and SHALL return an
4444 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4445 argument is less than the second argument according to the order relation specified for
4446 “http://www.w3.org/2001/XMLSchema#dateTime” by [XS, part 2, section 3.2.7]. Otherwise, it
4447 SHALL return “False”. Note: if a dateTime value does not include a time-zone value, then an
4448 implicit time-zone value SHALL be assigned, as described in [XS].

- 4449 • urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal

4450 This function SHALL take two arguments of data-type “http://www.w3.org/2001/XMLSchema#
4451 dateTime” and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL
4452 return “True” if and only if the first argument is less than or equal to the second argument
4453 according to the order relation specified for “http://www.w3.org/2001/XMLSchema#dateTime” by
4454 [XS] part 2, section 3.2.7. Otherwise, it SHALL return “False”. Note: if a dateTime value does
4455 not include a time-zone value, then an implicit time-zone value SHALL be assigned, as described
4456 in [XS].

- 4457 • urn:oasis:names:tc:xacml:1.0:function:date-greater-than

4458 This function SHALL take two arguments of data-type
4459 “http://www.w3.org/2001/XMLSchema#date” and SHALL return an
4460 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4461 argument is greater than the second argument according to the order relation specified for
4462 “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9. Otherwise, it SHALL
4463 return “False”. Note: if a date value does not include a time-zone value, then an implicit time-
4464 zone value SHALL be assigned, as described in [XS].

- 4465 • urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal

4466 This function SHALL take two arguments of data-type
4467 “http://www.w3.org/2001/XMLSchema#date” and SHALL return an
4468 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
4469 argument is greater than or equal to the second argument according to the order relation
4470 specified for “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9.

- 4471 Otherwise, it SHALL return “False”. Note: if a date value does not include a time-zone value,
4472 then an implicit time-zone value SHALL be assigned, as described in [XS].
- 4473 • urn:oasis:names:tc:xacml:1.0:function:date-less-than
 - 4474 This function SHALL take two arguments of data-type
 - 4475 “http://www.w3.org/2001/XMLSchema#date” and SHALL return an
 - 4476 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
 - 4477 argument is less than the second argument according to the order relation specified for
 - 4478 “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9. Otherwise, it SHALL
 - 4479 return “False”. Note: if a date value does not include a time-zone value, then an implicit time-
 - 4480 zone value SHALL be assigned, as described in [XS].
 - 4481 • urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal
 - 4482 This function SHALL take two arguments of data-type
 - 4483 “http://www.w3.org/2001/XMLSchema#date” and SHALL return an
 - 4484 “http://www.w3.org/2001/XMLSchema#boolean”. It SHALL return “True” if and only if the first
 - 4485 argument is less than or equal to the second argument according to the order relation specified
 - 4486 for “http://www.w3.org/2001/XMLSchema#date” by [XS] part 2, section 3.2.9. Otherwise, it
 - 4487 SHALL return “False”. Note: if a date value does not include a time-zone value, then an implicit
 - 4488 time-zone value SHALL be assigned, as described in [XS].

4489 **A.3.9 String functions**

4490 The following functions operate on strings and convert to and from other data types.

- 4491 • urn:oasis:names:tc:xacml:2.0:function:string-concatenate
 - 4492 This function SHALL take two or more arguments of data-type
 - 4493 “http://www.w3.org/2001/XMLSchema#string” and SHALL return a
 - 4494 “http://www.w3.org/2001/XMLSchema#string”. The result SHALL be the concatenation, in order,
 - 4495 of the arguments.
- 4496 • urn:oasis:names:tc:xacml:3.0:function:boolean-from-string
 - 4497 This function SHALL take one argument of data-type
 - 4498 “http://www.w3.org/2001/XMLSchema#string”, and SHALL return an
 - 4499 “http://www.w3.org/2001/XMLSchema#boolean”. The result SHALL be the string converted to a
 - 4500 boolean. If the argument is not a valid lexical representation of a boolean, then the result SHALL
 - 4501 be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4502 • urn:oasis:names:tc:xacml:3.0:function:string-from-boolean
 - 4503 This function SHALL take one argument of data-type
 - 4504 “http://www.w3.org/2001/XMLSchema#boolean”, and SHALL return an
 - 4505 “http://www.w3.org/2001/XMLSchema#string”. The result SHALL be the boolean converted to a
 - 4506 string in the canonical form specified in [XS].
- 4507 • urn:oasis:names:tc:xacml:3.0:function:integer-from-string
 - 4508 This function SHALL take one argument of data-type
 - 4509 “http://www.w3.org/2001/XMLSchema#string”, and SHALL return an
 - 4510 “http://www.w3.org/2001/XMLSchema#integer”. The result SHALL be the string converted to an
 - 4511 integer. If the argument is not a valid lexical representation of an integer, then the result SHALL
 - 4512 be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4513 • urn:oasis:names:tc:xacml:3.0:function:string-from-integer
 - 4514 This function SHALL take one argument of data-type
 - 4515 “http://www.w3.org/2001/XMLSchema#integer”, and SHALL return an
 - 4516 “http://www.w3.org/2001/XMLSchema#string”. The result SHALL be the integer converted to a
 - 4517 string in the canonical form specified in [XS].
- 4518 • urn:oasis:names:tc:xacml:3.0:function:double-from-string

- 4519 This function SHALL take one argument of data-type
 4520 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4521 "http://www.w3.org/2001/XMLSchema#double". The result SHALL be the string converted to a
 4522 double. If the argument is not a valid lexical representation of a double, then the result SHALL be
 4523 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4524 • urn:oasis:names:tc:xacml:3.0:function:string-from-double

4525 This function SHALL take one argument of data-type
 4526 "http://www.w3.org/2001/XMLSchema#double", and SHALL return an
 4527 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the double converted to a
 4528 string in the canonical form specified in **[XS]**.
 - 4529 • urn:oasis:names:tc:xacml:3.0:function:time-from-string

4530 This function SHALL take one argument of data-type
 4531 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4532 "http://www.w3.org/2001/XMLSchema#time". The result SHALL be the string converted to a time.
 4533 If the argument is not a valid lexical representation of a time, then the result SHALL be
 4534 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4535 • urn:oasis:names:tc:xacml:3.0:function:string-from-time

4536 This function SHALL take one argument of data-type
 4537 "http://www.w3.org/2001/XMLSchema#time", and SHALL return an
 4538 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the time converted to a
 4539 string in the canonical form specified in **[XS]**.
 - 4540 • urn:oasis:names:tc:xacml:3.0:function:date-from-string

4541 This function SHALL take one argument of data-type
 4542 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4543 "http://www.w3.org/2001/XMLSchema#date". The result SHALL be the string converted to a
 4544 date. If the argument is not a valid lexical representation of a date, then the result SHALL be
 4545 Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4546 • urn:oasis:names:tc:xacml:3.0:function:string-from-date

4547 This function SHALL take one argument of data-type
 4548 "http://www.w3.org/2001/XMLSchema#date", and SHALL return an
 4549 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the date converted to a
 4550 string in the canonical form specified in **[XS]**.
 - 4551 • urn:oasis:names:tc:xacml:3.0:function:dateFrom-string

4552 This function SHALL take one argument of data-type
 4553 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4554 "http://www.w3.org/2001/XMLSchema#dateFrom". The result SHALL be the string converted to a
 4555 dateFrom. If the argument is not a valid lexical representation of a dateFrom, then the result
 4556 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4557 urn:oasis:names:tc:xacml:3.0:function:string-from-dateFrom

4558 This function SHALL take one argument of data-type
 4559 "http://www.w3.org/2001/XMLSchema#dateFrom", and SHALL return an
 4560 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the dateFrom converted to a
 4561 string in the canonical form specified in **[XS]**.
 - 4562 • urn:oasis:names:tc:xacml:3.0:function:anyURI-from-string

4563 This function SHALL take one argument of data-type
 4564 "http://www.w3.org/2001/XMLSchema#string", and SHALL return a
 4565 "http://www.w3.org/2001/XMLSchema#anyURI". The result SHALL be the URI constructed by
 4566 converting the argument to an URI. If the argument is not a valid lexical representation of a URI,
 4567 then the result SHALL be Indeterminate with status code
 4568 urn:oasis:names:tc:xacml:1.0:status:syntax-error.

- 4569 • urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI
 4570 This function SHALL take one argument of data-type
 4571 "http://www.w3.org/2001/XMLSchema#anyURI", and SHALL return an
 4572 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the URI converted to a
 4573 string in the form it was originally represented in XML form.
- 4574 • urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-from-string
 4575 This function SHALL take one argument of data-type
 4576 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4577 "http://www.w3.org/2001/XMLSchema#dayTimeDuration ". The result SHALL be the string
 4578 converted to a dayTimeDuration. If the argument is not a valid lexical representation of a
 4579 dayTimeDuration, then the result SHALL be Indeterminate with status code
 4580 urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4581 • urn:oasis:names:tc:xacml:3.0:function:string-from-dayTimeDuration
 4582 This function SHALL take one argument of data-type
 4583 "http://www.w3.org/2001/XMLSchema#dayTimeDuration ", and SHALL return an
 4584 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the dayTimeDuration
 4585 converted to a string in the canonical form specified in **[XPathFunc]**.
- 4586 • urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-from-string
 4587 This function SHALL take one argument of data-type
 4588 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4589 "http://www.w3.org/2001/XMLSchema#yearMonthDuration". The result SHALL be the string
 4590 converted to a yearMonthDuration. If the argument is not a valid lexical representation of a
 4591 yearMonthDuration, then the result SHALL be Indeterminate with status code
 4592 urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4593 • urn:oasis:names:tc:xacml:3.0:function:string-from-yearMonthDuration
 4594 This function SHALL take one argument of data-type
 4595 "http://www.w3.org/2001/XMLSchema#yearMonthDuration", and SHALL return an
 4596 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the yearMonthDuration
 4597 converted to a string in the canonical form specified in **[XPathFunc]**.
- 4598 • urn:oasis:names:tc:xacml:3.0:function:x500Name-from-string
 4599 This function SHALL take one argument of data-type
 4600 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4601 "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". The result SHALL be the string converted
 4602 to an x500Name. If the argument is not a valid lexical representation of a X500Name, then the
 4603 result SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4604 • urn:oasis:names:tc:xacml:3.0:function:string-from-x500Name
 4605 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:1.0:data-
 4606 type:x500Name", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result
 4607 SHALL be the x500Name converted to a string in the form it was originally represented in XML
 4608 form..
- 4609 • urn:oasis:names:tc:xacml:3.0:function:rfc822Name-from-string
 4610 This function SHALL take one argument of data-type
 4611 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
 4612 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". The result SHALL be the string converted
 4613 to an rfc822Name. If the argument is not a valid lexical representation of an rfc822Name, then the
 4614 result SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
- 4615 • urn:oasis:names:tc:xacml:3.0:function:string-from-rfc822Name
 4616 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:1.0:data-
 4617 type:rfc822Name", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The

- 4618 result SHALL be the rfc822Name converted to a string in the form it was originally represented in
4619 XML form.
- 4620 • urn:oasis:names:tc:xacml:3.0:function:ipAddress-from-string
4621 This function SHALL take one argument of data-type
4622 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
4623 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". The result SHALL be the string converted to
4624 an ipAddress. If the argument is not a valid lexical representation of an ipAddress, then the result
4625 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4626 • urn:oasis:names:tc:xacml:3.0:function:string-from-ipAddress
4627 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:2.0:data-
4628 type:ipAddress", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result
4629 SHALL be the ipAddress converted to a string in the form it was originally represented in XML
4630 form.
 - 4631 • urn:oasis:names:tc:xacml:3.0:function:dnsName-from-string
4632 This function SHALL take one argument of data-type
4633 "http://www.w3.org/2001/XMLSchema#string", and SHALL return an
4634 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". The result SHALL be the string converted to
4635 a dnsName. If the argument is not a valid lexical representation of a dnsName, then the result
4636 SHALL be Indeterminate with status code urn:oasis:names:tc:xacml:1.0:status:syntax-error.
 - 4637 • urn:oasis:names:tc:xacml:3.0:function:string-from-dnsName
4638 This function SHALL take one argument of data-type "urn:oasis:names:tc:xacml:2.0:data-
4639 type:dnsName", and SHALL return an "http://www.w3.org/2001/XMLSchema#string". The result
4640 SHALL be the dnsName converted to a string in the form it was originally represented in XML
4641 form.
 - 4642 • urn:oasis:names:tc:xacml:3.0:function:string-starts-with
4643 This function SHALL take two arguments of data-type
4644 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a
4645 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string
4646 begins with the first string, and false otherwise. Equality testing SHALL be done as defined for
4647 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4648 • urn:oasis:names:tc:xacml:3.0:function:anyURI-starts-with
4649 This function SHALL take a first argument of data-
4650 type"http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type
4651 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a
4652 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted
4653 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI begins with the string,
4654 and false otherwise. Equality testing SHALL be done as defined for
4655 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4656 • urn:oasis:names:tc:xacml:3.0:function:string-ends-with
4657 This function SHALL take two arguments of data-type
4658 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a
4659 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string
4660 ends with the first string, and false otherwise. Equality testing SHALL be done as defined for
4661 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4662 • urn:oasis:names:tc:xacml:3.0:function:anyURI-ends-with
4663 This function SHALL take a first argument of data-type
4664 "http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type
4665 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a
4666 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted
4667 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI ends with the string,

- 4668 and false otherwise. Equality testing SHALL be done as defined for
4669 urn:oasis:names:tc:xacml:1.0:function:string-equal.
- 4670 • urn:oasis:names:tc:xacml:3.0:function:string-contains
4671 This function SHALL take two arguments of data-type
4672 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a
4673 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the second string
4674 contains the first string, and false otherwise. Equality testing SHALL be done as defined for
4675 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4676 • urn:oasis:names:tc:xacml:3.0:function:anyURI-contains
4677 This function SHALL take a first argument of data-type
4678 "http://www.w3.org/2001/XMLSchema#string" and an a second argument of data-type
4679 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return a
4680 "http://www.w3.org/2001/XMLSchema#boolean". The result SHALL be true if the URI converted
4681 to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI contains the string, and
4682 false otherwise. Equality testing SHALL be done as defined for
4683 urn:oasis:names:tc:xacml:1.0:function:string-equal.
 - 4684 • urn:oasis:names:tc:xacml:3.0:function:string-substring
4685 This function SHALL take a first argument of data-type
4686 "http://www.w3.org/2001/XMLSchema#string" and a second and a third argument of type
4687 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return a
4688 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the substring of the first
4689 argument beginning at the position given by the second argument and ending at the position
4690 before the position given by the third argument. The first character of the string has position zero.
4691 The negative integer value -1 given for the third arguments indicates the end of the string. If the
4692 second or third arguments are out of bounds, then the function MUST evaluate to Indeterminate
4693 with a status code of urn:oasis:names:tc:xacml:1.0:status:processing-error.
 - 4694 • urn:oasis:names:tc:xacml:3.0:function:anyURI-substring
4695 This function SHALL take a first argument of data-type
4696 "http://www.w3.org/2001/XMLSchema#anyURI" and a second and a third argument of type
4697 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return a
4698 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the substring of the first
4699 argument converted to a string with urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI
4700 beginning at the position given by the second argument and ending at the position before the
4701 position given by the third argument. The first character of the URI converted to a string has
4702 position zero. The negative integer value -1 given for the third arguments indicates the end of the
4703 string. If the second or third arguments are out of bounds, then the function MUST evaluate to
4704 Indeterminate with a status code of
4705 urn:oasis:names:tc:xacml:1.0:status:processing-error. If the resulting substring
4706 is not syntactically a valid URI, then the function MUST evaluate to Indeterminate with a status
4707 code of urn:oasis:names:tc:xacml:1.0:status:processing-error.

4708

4709 **A.3.10 Bag functions**

4710 These functions operate on a **bag** of 'type' values, where type is one of the primitive data-types, and x.x
4711 is a version of XACML where the function has been defined. Some additional conditions defined for
4712 each function below SHALL cause the expression to evaluate to "Indeterminate".

- 4713 • urn:oasis:names:tc:xacml:x.x:function:type-one-and-only
4714 This function SHALL take a **bag** of 'type' values as an argument and SHALL return a value of
4715 'type'. It SHALL return the only value in the **bag**. If the **bag** does not have one and only one
4716 value, then the expression SHALL evaluate to "Indeterminate".

- 4717 • urn:oasis:names:tc:xacml:x.x:function:type-bag-size
- 4718 This function SHALL take a **bag** of 'type' values as an argument and SHALL return an
- 4719 "http://www.w3.org/2001/XMLSchema#integer" indicating the number of values in the **bag**.
- 4720 • urn:oasis:names:tc:xacml:x.x:function:type-is-in
- 4721 This function SHALL take an argument of 'type' as the first argument and a **bag** of 'type' values
- 4722 as the second argument and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean".
- 4723 The function SHALL evaluate to "True" if and only if the first argument matches by the
- 4724 "urn:oasis:names:tc:xacml:x.x:function:type-equal" any value in the **bag**. Otherwise, it SHALL
- 4725 return "False".
- 4726 • urn:oasis:names:tc:xacml:x.x:function:type-bag
- 4727 This function SHALL take any number of arguments of 'type' and return a **bag** of 'type' values
- 4728 containing the values of the arguments. An application of this function to zero arguments SHALL
- 4729 produce an empty **bag** of the specified data-type.

4730 A.3.11 Set functions

4731 These functions operate on **bags** mimicking sets by eliminating duplicate elements from a **bag**.

- 4732 • urn:oasis:names:tc:xacml:x.x:function:type-intersection
- 4733 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4734 **bag** of 'type' values such that it contains only elements that are common between the two **bags**,
- 4735 which is determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal". No duplicates, as
- 4736 determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal", SHALL exist in the result.
- 4737 • urn:oasis:names:tc:xacml:x.x:function:type-at-least-one-member-of
- 4738 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4739 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL evaluate to "True" if and
- 4740 only if at least one element of the first argument is contained in the second argument as
- 4741 determined by "urn:oasis:names:tc:xacml:x.x:function:type-is-in".
- 4742 • urn:oasis:names:tc:xacml:x.x:function:type-union
- 4743 This function SHALL take two or more arguments that are both a **bag** of 'type' values. The
- 4744 expression SHALL return a **bag** of 'type' such that it contains all elements of all the argument
- 4745 **bags**. No duplicates, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal",
- 4746 SHALL exist in the result.
- 4747 • urn:oasis:names:tc:xacml:x.x:function:type-subset
- 4748 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4749 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the first
- 4750 argument is a subset of the second argument. Each argument SHALL be considered to have had
- 4751 its duplicates removed, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal",
- 4752 before the subset calculation.
- 4753 • urn:oasis:names:tc:xacml:x.x:function:type-set-equals
- 4754 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL return a
- 4755 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return the result of applying
- 4756 "urn:oasis:names:tc:xacml:1.0:function:and" to the application of
- 4757 "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the first and second arguments and the
- 4758 application of "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the second and first
- 4759 arguments.

4760 A.3.12 Higher-order bag functions

4761 This section describes functions in XACML that perform operations on **bags** such that functions may be

4762 applied to the **bags** in general.

4763 • urn:oasis:names:tc:xacml:3.0:function:any-of

4764 This function applies a Boolean function between specific primitive values and a **bag** of values,
4765 and SHALL return "True" if and only if the **predicate** is "True" for at least one element of the **bag**.

4766 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL
4767 be an <Function> element that names a Boolean function that takes n arguments of primitive
4768 types. Under the remaining n arguments, n-1 parameters SHALL be values of primitive data-
4769 types and one SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated as
4770 if the function named in the <Function> argument were applied to the n-1 non-bag arguments
4771 and each element of the bag argument and the results are combined with
4772 "urn:oasis:names:tc:xacml:1.0:function:or".

4773 For example, the following expression SHALL return "True":

```
4774 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">  
4775   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>  
4776   <AttributeValue  
4777     DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4778   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4779     <AttributeValue  
4780       DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>  
4781     <AttributeValue  
4782       DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4783     <AttributeValue  
4784       DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>  
4785     <AttributeValue  
4786       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4787   </Apply>  
4788 </Apply>
```

4789 This expression is "True" because the first argument is equal to at least one of the elements of
4790 the **bag**, according to the function.

4791 • urn:oasis:names:tc:xacml:3.0:function:all-of

4792 This function applies a Boolean function between a specific primitive value and a **bag** of values,
4793 and returns "True" if and only if the **predicate** is "True" for every element of the **bag**.

4794 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL
4795 be a <Function> element that names a Boolean function that takes n arguments of primitive
4796 types. Under the remaining n arguments, n-1 parameters SHALL be values of primitive data-
4797 types and one SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated as
4798 if the function named in the <Function> argument were applied to the n-1 non-bag arguments
4799 and each element of the bag argument and the results are combined with
4800 "urn:oasis:names:tc:xacml:1.0:function:and".

4801 For example, the following expression SHALL evaluate to "True":

```
4802 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:all-of">  
4803   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4804     greater-than"/>  
4805   <AttributeValue  
4806     DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>  
4807   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4808     <AttributeValue  
4809       DataType="http://www.w3.org/2001/XMLSchema#integer">9</AttributeValue>  
4810     <AttributeValue  
4811       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>  
4812     <AttributeValue  
4813       DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>  
4814     <AttributeValue  
4815       DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>  
4816   </Apply>  
4817 </Apply>
```

4818 This expression is "True" because the first argument (10) is greater than all of the elements of the
4819 **bag** (9,3,4 and 2).

4820 • urn:oasis:names:tc:xacml:3.0:function:any-of-any

4821 This function applies a Boolean function on each tuple from the cross product on all bags
4822 arguments, and returns "True" if and only if the **predicate** is "True" for at least one inside-function
4823 call.

4824 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL
4825 be an <Function> element that names a Boolean function that takes n arguments. The
4826 remaining arguments are either primitive data types or bags of primitive types. The expression
4827 SHALL be evaluated as if the function named in the <Function> argument was applied between
4828 every tuple of the cross product on all bags and the primitive values, and the results were
4829 combined using "urn:oasis:names:tc:xacml:1.0:function:or". The semantics are that the result of
4830 the expression SHALL be "True" if and only if the applied **predicate** is "True" for at least one
4831 function call on the tuples from the **bags** and primitive values.

4832 For example, the following expression SHALL evaluate to "True":

```
4833 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of-any">  
4834   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>  
4835   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4836     <AttributeValue  
4837       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4838     <AttributeValue  
4839       DataType="http://www.w3.org/2001/XMLSchema#string">Mary</AttributeValue>  
4840   </Apply>  
4841   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4842     <AttributeValue  
4843       DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>  
4844     <AttributeValue  
4845       DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>  
4846     <AttributeValue  
4847       DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>  
4848     <AttributeValue  
4849       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>  
4850   </Apply>  
4851 </Apply>
```

4852 This expression is "True" because at least one of the elements of the first **bag**, namely "Ringo", is
4853 equal to at least one of the elements of the second **bag**.

4854 • urn:oasis:names:tc:xacml:1.0:function:all-of-any

4855 This function applies a Boolean function between the elements of two **bags**. The expression
4856 SHALL be "True" if and only if the supplied **predicate** is "True" between each element of the first
4857 **bag** and any element of the second **bag**.

4858 This function SHALL take three arguments. The first argument SHALL be an <Function>
4859 element that names a Boolean function that takes two arguments of primitive types. The second
4860 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a
4861 primitive data-type. The expression SHALL be evaluated as if the
4862 "urn:oasis:names:tc:xacml:3.0:function:any-of" function had been applied to each value of the first
4863 **bag** and the whole of the second **bag** using the supplied xacml:Function, and the results were
4864 then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

4865 For example, the following expression SHALL evaluate to "True":

```
4866 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-any">  
4867   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4868   greater-than"/>  
4869   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4870     <AttributeValue  
4871       DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
```

```

4872         <AttributeValue
4873         DataType="http://www.w3.org/2001/XMLSchema#integer">20</AttributeValue>
4874         </Apply>
4875         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4876         <AttributeValue
4877         DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4878         <AttributeValue
4879         DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4880         <AttributeValue
4881         DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4882         <AttributeValue
4883         DataType="http://www.w3.org/2001/XMLSchema#integer">19</AttributeValue>
4884         </Apply>
4885     </Apply>

```

4886 This expression is "True" because each of the elements of the first **bag** is greater than at least
4887 one of the elements of the second **bag**.

4888 • urn:oasis:names:tc:xacml:1.0:function:any-of-all

4889 This function applies a Boolean function between the elements of two **bags**. The expression
4890 SHALL be "True" if and only if the supplied **predicate** is "True" between each element of the
4891 second **bag** and any element of the first **bag**.

4892 This function SHALL take three arguments. The first argument SHALL be an <Function>
4893 element that names a Boolean function that takes two arguments of primitive types. The second
4894 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a
4895 primitive data-type. The expression SHALL be evaluated as if the
4896 "urn:oasis:names:tc:xacml:3.0:function:any-of" function had been applied to each value of the
4897 second **bag** and the whole of the first **bag** using the supplied xacml:Function, and the results
4898 were then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

4899 For example, the following expression SHALL evaluate to "True":

```

4900 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-all">
4901     <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-
4902     greater-than"/>
4903     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4904     <AttributeValue
4905     DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4906     <AttributeValue
4907     DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4908     </Apply>
4909     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4910     <AttributeValue
4911     DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4912     <AttributeValue
4913     DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
4914     <AttributeValue
4915     DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4916     <AttributeValue
4917     DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
4918     </Apply>
4919 </Apply>

```

4920 This expression is "True" because, for all of the values in the second **bag**, there is a value in the
4921 first **bag** that is greater.

4922 • urn:oasis:names:tc:xacml:1.0:function:all-of-all

4923 This function applies a Boolean function between the elements of two **bags**. The expression
4924 SHALL be "True" if and only if the supplied **predicate** is "True" between each and every element
4925 of the first **bag** collectively against all the elements of the second **bag**.

4926 This function SHALL take three arguments. The first argument SHALL be an <Function>
4927 element that names a Boolean function that takes two arguments of primitive types. The second

4928 argument SHALL be a **bag** of a primitive data-type. The third argument SHALL be a **bag** of a
4929 primitive data-type. The expression is evaluated as if the function named in the <Function>
4930 element were applied between every element of the second argument and every element of the
4931 third argument and the results were combined using "urn:oasis:names:tc:xacml:1.0:function:and".
4932 The semantics are that the result of the expression is "True" if and only if the applied **predicate** is
4933 "True" for all elements of the first **bag** compared to all the elements of the second **bag**.

4934 For example, the following expression SHALL evaluate to "True":

```
4935 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-all">  
4936   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-  
4937   greater-than"/>  
4938   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4939     <AttributeValue  
4940     DataType="http://www.w3.org/2001/XMLSchema#integer">6</AttributeValue>  
4941     <AttributeValue  
4942     DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>  
4943   </Apply>  
4944   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">  
4945     <AttributeValue  
4946     DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>  
4947     <AttributeValue  
4948     DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>  
4949     <AttributeValue  
4950     DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>  
4951     <AttributeValue  
4952     DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>  
4953   </Apply>  
4954 </Apply>
```

4955 This expression is "True" because all elements of the first **bag**, "5" and "6", are each greater than
4956 all of the integer values "1", "2", "3", "4" of the second **bag**.

4957 • urn:oasis:names:tc:xacml:3.0:function:map

4958 This function converts a **bag** of values to another **bag** of values.

4959 This function SHALL take n+1 arguments, where n is one or greater. The first argument SHALL
4960 be a <Function> element naming a function that takes a n arguments of a primitive data-type
4961 and returns a value of a primitive data-type Under the remaining n arguments, n-1 parameters
4962 SHALL be values of primitive data-types and one SHALL be a **bag** of a primitive data-type. The
4963 expression SHALL be evaluated as if the function named in the <Function> argument were
4964 applied to the n-1 non-bag arguments and each element of the bag argument and resulting in a
4965 **bag** of the converted value. The result SHALL be a **bag** of the primitive data-type that is returned
4966 by the function named in the <xacml:Function> element.

4967 For example, the following expression,

```
4968 <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:map">  
4969   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-  
4970   normalize-to-lower-case">  
4971   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">  
4972     <AttributeValue  
4973     DataType="http://www.w3.org/2001/XMLSchema#string">Hello</AttributeValue>  
4974     <AttributeValue  
4975     DataType="http://www.w3.org/2001/XMLSchema#string">World!</AttributeValue>  
4976   </Apply>  
4977 </Apply>
```

4978 evaluates to a **bag** containing "hello" and "world!".

4979 A.3.13 Regular-expression-based functions

4980 These functions operate on various types using regular expressions and evaluate to
4981 "http://www.w3.org/2001/XMLSchema#boolean".

- 4982 • urn:oasis:names:tc:xacml:1.0:function:string-regexp-match
- 4983 This function decides a regular expression match. It SHALL take two arguments of
4984 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an
4985 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
4986 expression and the second argument SHALL be a general string. The function specification
4987 SHALL be that of the "xf:matches" function with the arguments reversed [XF] Section 7.6.2.
- 4988 • urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match
- 4989 This function decides a regular expression match. It SHALL take two arguments; the first is of
4990 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
4991 "http://www.w3.org/2001/XMLSchema#anyURI". It SHALL return an
4992 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
4993 expression and the second argument SHALL be a URI. The function SHALL convert the second
4994 argument to type "http://www.w3.org/2001/XMLSchema#string" with
4995 urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI, then apply
4996 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 4997 • urn:oasis:names:tc:xacml:2.0:function:ipAddress-regexp-match
- 4998 This function decides a regular expression match. It SHALL take two arguments; the first is of
4999 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
5000 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". It SHALL return an
5001 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
5002 expression and the second argument SHALL be an IPv4 or IPv6 address. The function SHALL
5003 convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with
5004 urn:oasis:names:tc:xacml:3.0:function:string-from-ipAddress, then apply
5005 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5006 • urn:oasis:names:tc:xacml:2.0:function:dnsName-regexp-match
- 5007 This function decides a regular expression match. It SHALL take two arguments; the first is of
5008 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
5009 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". It SHALL return an
5010 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
5011 expression and the second argument SHALL be a DNS name. The function SHALL convert the
5012 second argument to type "http://www.w3.org/2001/XMLSchema#string" with
5013 urn:oasis:names:tc:xacml:3.0:function:string-from-dnsName, then apply
5014 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5015 • urn:oasis:names:tc:xacml:2.0:function:rfc822Name-regexp-match
- 5016 This function decides a regular expression match. It SHALL take two arguments; the first is of
5017 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
5018 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". It SHALL return an
5019 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
5020 expression and the second argument SHALL be an RFC 822 name. The function SHALL convert
5021 the second argument to type "http://www.w3.org/2001/XMLSchema#string" with
5022 urn:oasis:names:tc:xacml:3.0:function:string-from-rfc822Name, then apply
5023 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".
- 5024 • urn:oasis:names:tc:xacml:2.0:function:x500Name-regexp-match
- 5025 This function decides a regular expression match. It SHALL take two arguments; the first is of
5026 type "http://www.w3.org/2001/XMLSchema#string" and the second is of type
5027 "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". It SHALL return an
5028 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular
5029 expression and the second argument SHALL be an X.500 directory name. The function SHALL
5030 convert the second argument to type "http://www.w3.org/2001/XMLSchema#string" with
5031 urn:oasis:names:tc:xacml:3.0:function:string-from-x500Name, then apply
5032 "urn:oasis:names:tc:xacml:1.0:function:string-regexp-match".

5033 **A.3.14 Special match functions**

5034 These functions operate on various types and evaluate to
5035 "http://www.w3.org/2001/XMLSchema#boolean" based on the specified standard matching algorithm.

- 5036 • urn:oasis:names:tc:xacml:1.0:function:x500Name-match

5037 This function shall take two arguments of "urn:oasis:names:tc:xacml:1.0:data-type:x500Name"
5038 and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It shall return "True" if and
5039 only if the first argument matches some terminal sequence of RDNs from the second argument
5040 when compared using x500Name-equal.

5041 As an example (non-normative), if the first argument is "O=Medico Corp,C=US" and the second
5042 argument is "cn=John Smith,o=Medico Corp,c=US", then the function will return "True".

- 5043 • urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match

5044 This function SHALL take two arguments, the first is of data-type
5045 "http://www.w3.org/2001/XMLSchema#string" and the second is of data-type
5046 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" and SHALL return an
5047 "http://www.w3.org/2001/XMLSchema#boolean". This function SHALL evaluate to "True" if the
5048 first argument matches the second argument according to the following specification.

5049 An RFC822 name consists of a local-part followed by "@" followed by a domain-part. The local-
5050 part is case-sensitive, while the domain-part (which is usually a DNS name) is not case-sensitive.

5051 The second argument contains a complete rfc822Name. The first argument is a complete or
5052 partial rfc822Name used to select appropriate values in the second argument as follows.

5053 In order to match a particular address in the second argument, the first argument must specify the
5054 complete mail address to be matched. For example, if the first argument is
5055 "Anderson@sun.com", this matches a value in the second argument of "Anderson@sun.com"
5056 and "Anderson@SUN.COM", but not "Anne.Anderson@sun.com", "anderson@sun.com" or
5057 "Anderson@east.sun.com".

5058 In order to match any address at a particular domain in the second argument, the first argument
5059 must specify only a domain name (usually a DNS name). For example, if the first argument is
5060 "sun.com", this matches a value in the second argument of "Anderson@sun.com" or
5061 "Baxter@SUN.COM", but not "Anderson@east.sun.com".

5062 In order to match any address in a particular domain in the second argument, the first argument
5063 must specify the desired domain-part with a leading ".". For example, if the first argument is
5064 ".east.sun.com", this matches a value in the second argument of "Anderson@east.sun.com" and
5065 "anne.anderson@ISRG.EAST.SUN.COM" but not "Anderson@sun.com".

5066 **A.3.15 XPath-based functions**

5067 This section specifies functions that take XPath expressions for arguments. An XPath expression
5068 evaluates to a node-set, which is a set of XML nodes that match the expression. A node or node-set is
5069 not in the formal data-type system of XACML. All comparison or other operations on node-sets are
5070 performed in isolation of the particular function specified. The context nodes and namespace mappings
5071 of the XPath expressions are defined by the XPath data-type, see section B.3. The following functions
5072 are defined:

- 5073 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-count

5074 This function SHALL take an "urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression" as an
5075 argument and evaluates to an "http://www.w3.org/2001/XMLSchema#integer". The value
5076 returned from the function SHALL be the count of the nodes within the node-set that match the
5077 given XPath expression. If the <Content> element of the category to which the XPath
5078 expression applies to is not present in the request, this function SHALL return a value of zero.

- 5079 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-equal

5080 This function SHALL take two “urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression”
5081 arguments and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. The function
5082 SHALL return "True" if any of the XML nodes in the node-set matched by the first argument
5083 equals any of the XML nodes in the node-set matched by the second argument. Two nodes are
5084 considered equal if they have the same identity. If the <Content> element of the category to
5085 which either XPath expression applies to is not present in the request, this function SHALL return
5086 a value of “False”.

5087 • urn:oasis:names:tc:xacml:3.0:function:xpath-node-match

5088 This function SHALL take two “urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression”
5089 arguments and SHALL return an “http://www.w3.org/2001/XMLSchema#boolean”. This function
5090 SHALL evaluate to "True" if one of the following two conditions is satisfied: (1) Any of the XML
5091 nodes in the node-set matched by the first argument is equal to any of the XML nodes in the
5092 node-set matched by the second argument; (2) any node below any of the XML nodes in the
5093 node-set matched by the first argument is equal to any of the XML nodes in the node-set
5094 matched by the second argument. Two nodes are considered equal if they have the same
5095 identity. If the <Content> element of the category to which either XPath expression applies to is
5096 not present in the request, this function SHALL return a value of “False”.

5097 NOTE: The first **condition** is equivalent to "xpath-node-equal", and guarantees that "xpath-node-equal" is
5098 a special case of "xpath-node-match".

5099 A.3.16 Other functions

5100 • urn:oasis:names:tc:xacml:3.0:function:access-permitted

5101 This function SHALL take an “http://www.w3.org/2001/XMLSchema#anyURI” and an
5102 “http://www.w3.org/2001/XMLSchema#string” as arguments. The first argument SHALL be
5103 interpreted as an **attribute** category. The second argument SHALL be interpreted as the XML
5104 content of an <Attributes> element with *Category* equal to the first argument. The function
5105 evaluates to an “http://www.w3.org/2001/XMLSchema#boolean”. This function SHALL return
5106 "True" if and only if the **policy** evaluation described below returns the value of "Permit".

5107 The following evaluation is described as if the **context** is actually instantiated, but it is only
5108 required that an equivalent result be obtained.

5109 The function SHALL construct a new **context**, by copying all the information from the current
5110 **context**, omitting any <Attributes> element with *Category* equal to the first argument. The
5111 second function argument SHALL be added to the **context** as the content of an <Attributes>
5112 element with *Category* equal to the first argument.

5113 The function SHALL invoke a complete **policy** evaluation using the newly constructed **context**.
5114 This evaluation SHALL be completely isolated from the evaluation which invoked the function, but
5115 shall use all current **policies** and combining algorithms, including any per request **policies**.

5116 The **PDP** SHALL detect any loop which may occur if successive evaluations invoke this function
5117 by counting the number of total invocations of any instance of this function during any single initial
5118 invocation of the **PDP**. If the total number of invocations exceeds the bound for such invocations,
5119 the initial invocation of this function evaluates to Indeterminate with a
5120 “urn:oasis:names:tc:xacml:1.0:status:processing-error” status code. Also, see the security
5121 considerations in section 9.1.8.

5122 A.3.17 Extension functions and primitive types

5123 Functions and primitive types are specified by string identifiers allowing for the introduction of functions in
5124 addition to those specified by XACML. This approach allows one to extend the XACML module with
5125 special functions and special primitive data-types.

5126 In order to preserve the integrity of the XACML evaluation strategy, the result of an extension function
5127 SHALL depend only on the values of its arguments. Global and hidden parameters SHALL NOT affect

5128 the evaluation of an expression. Functions SHALL NOT have side effects, as evaluation order cannot be
5129 guaranteed in a standard way.

5130 **A.4 Functions, data types, attributes and algorithms planned for** 5131 **deprecation**

5132 The following functions, data types and algorithms have been defined by previous versions of XACML
5133 and newer and better alternatives are defined in XACML 3.0. Their use is discouraged for new use and
5134 they are candidates for deprecation in future versions of XACML.

5135 The following xpath based functions have been replaced with equivalent functions which use the new
5136 urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression datatype instead of strings.

- 5137 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-count
- 5138 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-count
- 5139 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal
- 5140 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-equal
- 5141 • urn:oasis:names:tc:xacml:1.0:function:xpath-node-match
- 5142 • Replaced with urn:oasis:names:tc:xacml:3.0:function:xpath-node-match

5143 The following URI and string concatenation function has been replaced with a string to URI conversion
5144 function, which allows the use of the general string functions with URI through string conversion.

- 5145 • urn:oasis:names:tc:xacml:2.0:function:uri-string-concatenate
- 5146 • Replaced by urn:oasis:names:tc:xacml:3.0:function:string-from-anyURI

5147 The following identifiers have been replaced with official identifiers defined by W3C.

- 5148 • <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration>
- 5149 • Replaced with <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- 5150 • <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration>
- 5151 • Replaced with <http://www.w3.org/2001/XMLSchema#yearMonthDuration>

5152 The following functions have been replaced with functions which use the updated dayTimeDuration and
5153 yearMonthDuration data types.

- 5154 • urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal
- 5155 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dayTimeDuration-equal
- 5156 • urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-equal
- 5157 • Replaced with urn:oasis:names:tc:xacml:3.0:function:yearMonthDuration-equal
- 5158 • urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration
- 5159 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-add-dayTimeDuration
- 5160 • urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration
- 5161 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-add-yearMonthDuration
- 5162 • urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration
- 5163 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-dayTimeDuration
- 5164 • urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-yearMonthDuration
- 5165 • Replaced with urn:oasis:names:tc:xacml:3.0:function:dateTime-subtract-yearMonthDuration
- 5166 • urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration
- 5167 • Replaced with urn:oasis:names:tc:xacml:3.0:function:date-add-yearMonthDuration
- 5168 • urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration
- 5169 • Replaced with urn:oasis:names:tc:xacml:3.0:function:date-subtract-yearMonthDuration

- 5170 The following attribute identifiers have been replaced with new identifiers
- 5171 • urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address
- 5172 • Replaced with urn:oasis:names:tc:xacml:3.0:subject:authn-locality:ip-
- 5173 address
- 5174 • urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name
- 5175 • Replaced with urn:oasis:names:tc:xacml:3.0:subject:authn-
- 5176 locality:dns-name
- 5177
- 5178 The following combining algorithms have been replaced with new variants which allow for better handling
- 5179 of “Indeterminate” results.
- 5180 • urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides
- 5181 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides
- 5182 • urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides
- 5183 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides
- 5184 • urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides
- 5185 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides
- 5186 • urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides
- 5187 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides
- 5188 • urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides
- 5189 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides
- 5190 • urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides
- 5191 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-overrides
- 5192 • urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-overrides
- 5193 • Replaced with urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-overrides
- 5194 • urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-overrides
- 5195 • Replaced with urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-overrides

5196 Appendix B. XACML identifiers (normative)

5197 This section defines standard identifiers for commonly used entities.

5198 B.1 XACML namespaces

5199 XACML is defined using this identifier.

5200 `urn:oasis:names:tc:xacml:3.0:core:schema`

5201 B.2 Attribute categories

5202 The following **attribute** category identifiers MUST be used when an XACML 2.0 or earlier **policy** or
5203 request is translated into XACML 3.0.

5204 **Attributes** previously placed in the **Resource**, **Action**, and **Environment** sections of a request are
5205 placed in an **attribute** category with the following identifiers respectively. It is RECOMMENDED that they
5206 are used to list **attributes of resources**, **actions**, and the **environment** respectively when authoring
5207 XACML 3.0 **policies** or requests.

5208 `urn:oasis:names:tc:xacml:3.0:attribute-category:resource`

5209 `urn:oasis:names:tc:xacml:3.0:attribute-category:action`

5210 `urn:oasis:names:tc:xacml:3.0:attribute-category:environment`

5211 **Attributes** previously placed in the **Subject** section of a request are placed in an **attribute** category
5212 which is identical of the **subject** category in XACML 2.0, as defined below. It is RECOMMENDED that
5213 they are used to list **attributes of subjects** when authoring XACML 3.0 **policies** or requests.

5214 This identifier indicates the system entity that initiated the **access** request. That is, the initial entity in a
5215 request chain. If **subject** category is not specified in XACML 2.0, this is the default translation value.

5216 `urn:oasis:names:tc:xacml:1.0:subject-category:access-subject`

5217 This identifier indicates the system entity that will receive the results of the request (used when it is
5218 distinct from the access-**subject**).

5219 `urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject`

5220 This identifier indicates a system entity through which the **access** request was passed.

5221 `urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject`

5222 This identifier indicates a system entity associated with a local or remote codebase that generated the
5223 request. Corresponding **subject attributes** might include the URL from which it was loaded and/or the
5224 identity of the code-signer.

5225 `urn:oasis:names:tc:xacml:1.0:subject-category:codebase`

5226 This identifier indicates a system entity associated with the computer that initiated the **access** request.
5227 An example would be an IPsec identity.

5228 `urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine`

5229 B.3 Data-types

5230 The following identifiers indicate data-types that are defined in Section A.2.

5231 `urn:oasis:names:tc:xacml:1.0:data-type:x500Name`.

5232 `urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name`

5233 `urn:oasis:names:tc:xacml:2.0:data-type:ipAddress`

5234 `urn:oasis:names:tc:xacml:2.0:data-type:dnsName`

5235 `urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression`

- 5236 The following data-type identifiers are defined by XML Schema [XS].
- 5237 <http://www.w3.org/2001/XMLSchema#string>
- 5238 <http://www.w3.org/2001/XMLSchema#boolean>
- 5239 <http://www.w3.org/2001/XMLSchema#integer>
- 5240 <http://www.w3.org/2001/XMLSchema#double>
- 5241 <http://www.w3.org/2001/XMLSchema#time>
- 5242 <http://www.w3.org/2001/XMLSchema#date>
- 5243 <http://www.w3.org/2001/XMLSchema#dateTime>
- 5244 <http://www.w3.org/2001/XMLSchema#anyURI>
- 5245 <http://www.w3.org/2001/XMLSchema#hexBinary>
- 5246 <http://www.w3.org/2001/XMLSchema#base64Binary>
- 5247 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.
- 5248 <http://www.w3.org/2001/XMLSchema#dayTimeDuration>
- 5249 <http://www.w3.org/2001/XMLSchema#yearMonthDuration>

5251 B.4 Subject attributes

- 5252 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).
- 5253
- 5254 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.
- 5255
- 5256 This identifier indicates the name of the **subject**.
- 5257 `urn:oasis:names:tc:xacml:1.0:subject:subject-id`
- 5258 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.
- 5259
- 5260 `urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier`
- 5261 This identifier indicates a public key used to confirm the **subject's** identity.
- 5262 `urn:oasis:names:tc:xacml:1.0:subject:key-info`
- 5263 This identifier indicates the time at which the **subject** was authenticated.
- 5264 `urn:oasis:names:tc:xacml:1.0:subject:authentication-time`
- 5265 This identifier indicates the method used to authenticate the **subject**.
- 5266 `urn:oasis:names:tc:xacml:1.0:subject:authentication-method`
- 5267 This identifier indicates the time at which the **subject** initiated the **access** request, according to the **PEP**.
- 5268 `urn:oasis:names:tc:xacml:1.0:subject:request-time`
- 5269 This identifier indicates the time at which the **subject's** current session began, according to the **PEP**.
- 5270 `urn:oasis:names:tc:xacml:1.0:subject:session-start-time`
- 5271 The following identifiers indicate the location where authentication credentials were activated.
- 5272 This identifier indicates that the location is expressed as an IP address.
- 5273 `urn:oasis:names:tc:xacml:3.0:subject:authn-locality:ip-address`
- 5274 The corresponding **attribute** SHALL be of data-type "`urn:oasis:names:tc:xacml:2.0:data-type:ipAddress`".
- 5275 This identifier indicates that the location is expressed as a DNS name.
- 5276 `urn:oasis:names:tc:xacml:3.0:subject:authn-locality:dns-name`
- 5277 The corresponding **attribute** SHALL be of data-type "`urn:oasis:names:tc:xacml:2.0:data-type:dnsName`".

5278 Where a suitable **attribute** is already defined in LDAP [LDAP-1], [LDAP-2], the XACML identifier SHALL
5279 be formed by adding the **attribute** name to the URI of the LDAP specification. For example, the **attribute**
5280 name for the userPassword defined in the RFC 2256 SHALL be:
5281 `http://www.ietf.org/rfc/rfc2256.txt#userPassword`

5282 B.5 Resource attributes

5283 These identifiers indicate **attributes** of the **resource**. When used, it is RECOMMENDED they appear
5284 within the <Attributes> element of the request **context** with Category
5285 `urn:oasis:names:tc:xacml:3.0:attribute-category:resource`.

5286 This **attribute** identifies the **resource** to which **access** is requested.

5287 `urn:oasis:names:tc:xacml:1.0:resource:resource-id`

5288 This **attribute** identifies the namespace of the top element(s) of the contents of the <Content> element.
5289 In the case where the **resource** content is supplied in the request **context** and the **resource**
5290 namespaces are defined in the **resource**, the **PEP** MAY provide this **attribute** in the request to indicate
5291 the namespaces of the **resource** content. In this case there SHALL be one value of this **attribute** for
5292 each unique namespace of the top level elements in the <Content> element. The type of the
5293 corresponding **attribute** SHALL be “`http://www.w3.org/2001/XMLSchema#anyURI`”.

5294 `urn:oasis:names:tc:xacml:2.0:resource:target-namespace`

5295 B.6 Action attributes

5296 These identifiers indicate **attributes** of the **action** being requested. When used, it is RECOMMENDED
5297 they appear within the <Attributes> element of the request **context** with Category
5298 `urn:oasis:names:tc:xacml:3.0:attribute-category:action`.

5299 This **attribute** identifies the **action** for which **access** is requested.

5300 `urn:oasis:names:tc:xacml:1.0:action:action-id`

5301 Where the **action** is implicit, the value of the action-id **attribute** SHALL be

5302 `urn:oasis:names:tc:xacml:1.0:action:implied-action`

5303 This **attribute** identifies the namespace in which the action-id **attribute** is defined.

5304 `urn:oasis:names:tc:xacml:1.0:action:action-namespace`

5305 B.7 Environment attributes

5306 These identifiers indicate **attributes** of the **environment** within which the **decision request** is to be
5307 evaluated. When used in the **decision request**, it is RECOMMENDED they appear in the
5308 <Attributes> element of the request **context** with Category `urn:oasis:names:tc:xacml:3.0:attribute-`
5309 `category:environment`.

5310 This identifier indicates the current time at the **context handler**. In practice it is the time at which the
5311 request **context** was created. For this reason, if these identifiers appear in multiple places within a
5312 <Policy> or <PolicySet>, then the same value SHALL be assigned to each occurrence in the
5313 evaluation procedure, regardless of how much time elapses between the processing of the occurrences.

5314 `urn:oasis:names:tc:xacml:1.0:environment:current-time`

5315 The corresponding **attribute** SHALL be of data-type “`http://www.w3.org/2001/XMLSchema#time`”.

5316 `urn:oasis:names:tc:xacml:1.0:environment:current-date`

5317 The corresponding **attribute** SHALL be of data-type “`http://www.w3.org/2001/XMLSchema#date`”.

5318 `urn:oasis:names:tc:xacml:1.0:environment:current-dateTime`

5319 The corresponding **attribute** SHALL be of data-type “`http://www.w3.org/2001/XMLSchema#dateTime`”.

5320 **B.8 Status codes**

5321 The following status code values are defined.

5322 This identifier indicates success.

5323 urn:oasis:names:tc:xacml:1.0:status:ok

5324 This identifier indicates that all the **attributes** necessary to make a **policy decision** were not available (see Section 5.58).

5326 urn:oasis:names:tc:xacml:1.0:status:missing-attribute

5327 This identifier indicates that some **attribute** value contained a syntax error, such as a letter in a numeric field.

5329 urn:oasis:names:tc:xacml:1.0:status:syntax-error

5330 This identifier indicates that an error occurred during **policy** evaluation. An example would be division by zero.

5332 urn:oasis:names:tc:xacml:1.0:status:processing-error

5333 **B.9 Combining algorithms**

5334 The deny-overrides **rule-combining algorithm** has the following value for the ruleCombiningAlgId attribute:

5336 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides

5337 The deny-overrides **policy-combining algorithm** has the following value for the

5338 policyCombiningAlgId attribute:

5339 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides

5340 The permit-overrides **rule-combining algorithm** has the following value for the ruleCombiningAlgId attribute:

5342 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides

5343 The permit-overrides **policy-combining algorithm** has the following value for the

5344 policyCombiningAlgId attribute:

5345 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides

5346 The first-applicable **rule-combining algorithm** has the following value for the ruleCombiningAlgId attribute:

5348 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable

5349 The first-applicable **policy-combining algorithm** has the following value for the

5350 policyCombiningAlgId attribute:

5351 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable

5352 The only-one-applicable-policy **policy-combining algorithm** has the following value for the

5353 policyCombiningAlgId attribute:

5354 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable

5355 The ordered-deny-overrides **rule-combining algorithm** has the following value for the

5356 ruleCombiningAlgId attribute:

5357 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides

5358 The ordered-deny-overrides **policy-combining algorithm** has the following value for the

5359 policyCombiningAlgId attribute:

5360 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-

5361 overrides

5362 The ordered-permit-overrides **rule-combining algorithm** has the following value for the

5363 ruleCombiningAlgId attribute:

5364 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-
5365 overrides

5366 The ordered-permit-overrides **policy-combining algorithm** has the following value for the
5367 policyCombiningAlgId attribute:

5368 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-
5369 overrides

5370 The deny-unless-permit **rule-combining algorithm** has the following value for the
5371 policyCombiningAlgId attribute:

5372 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit

5373 The permit-unless-deny **rule-combining algorithm** has the following value for the
5374 policyCombiningAlgId attribute:

5375 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-unless-deny

5376 The deny-unless-permit **policy-combining algorithm** has the following value for the
5377 policyCombiningAlgId attribute:

5378 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit

5379 The permit-unless-deny **policy-combining algorithm** has the following value for the
5380 policyCombiningAlgId attribute:

5381 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-unless-deny

5382 The legacy deny-overrides **rule-combining algorithm** has the following value for the
5383 ruleCombiningAlgId attribute:

5384 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides

5385 The legacy deny-overrides **policy-combining algorithm** has the following value for the
5386 policyCombiningAlgId attribute:

5387 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides

5388 The legacy permit-overrides **rule-combining algorithm** has the following value for the
5389 ruleCombiningAlgId attribute:

5390 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides

5391 The legacy permit-overrides **policy-combining algorithm** has the following value for the
5392 policyCombiningAlgId attribute:

5393 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides

5394 The legacy ordered-deny-overrides **rule-combining algorithm** has the following value for the
5395 ruleCombiningAlgId attribute:

5396 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides

5397 The legacy ordered-deny-overrides **policy-combining algorithm** has the following value for the
5398 policyCombiningAlgId attribute:

5399 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-
5400 overrides

5401 The legacy ordered-permit-overrides **rule-combining algorithm** has the following value for the
5402 ruleCombiningAlgId attribute:

5403 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-
5404 overrides

5405 The legacy ordered-permit-overrides **policy-combining algorithm** has the following value for the
5406 policyCombiningAlgId attribute:

5407 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-
5408 overrides

5409

5410

Appendix C. Combining algorithms (normative)

5411 This section contains a description of the **rule-** and **policy-combining algorithms** specified by XACML.
5412 Pseudo code is normative, descriptions in English are non-normative.

5413 The legacy **combining algorithms** are defined in previous versions of XACML, and are retained for
5414 compatibility reasons. It is RECOMMENDED that the new **combining algorithms** are used instead of the
5415 legacy **combining algorithms** for new use.

5416 Note that in each case an implementation is conformant as long as it produces the same result as is
5417 specified here, regardless of how and in what order the implementation behaves internally.

5418 C.1 Extended Indeterminate values

5419 Some combining algorithms are defined in terms of an extended set of "Indeterminate" values. See
5420 section 7.10 for the definition of the Extended Indeterminate values. For these algorithms, the **PDP** MUST
5421 keep track of the extended set of "Indeterminate" values during **rule** and **policy** combining.

5422 The output of a combining algorithm which does not track the extended set of "Indeterminate" values
5423 MUST be treated as "Indeterminate{DP}" for the value "Indeterminate" by a combining algorithm which
5424 tracks the extended set of "Indeterminate" values.

5425 A combining algorithm which does not track the extended set of "Indeterminate" values MUST treat the
5426 output of a combining algorithm which tracks the extended set of "Indeterminate" values as an
5427 "Indeterminate" for any of the possible values of the extended set of "Indeterminate".

5428 C.2 Deny-overrides

5429 This section defines the "Deny-overrides" **rule-combining algorithm** of a **policy** and **policy-combining**
5430 **algorithm** of a **policy set**.

5431 This **combining algorithm** makes use of the extended "Indeterminate".

5432 The **rule combining algorithm** defined here has the following identifier:

5433 `urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides`

5434 The **policy combining algorithm** defined here has the following identifier:

5435 `urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-overrides`

5436 The following is a non-normative informative description of this **combining algorithm**.

5437 The deny overrides **combining algorithm** is intended for those cases where a deny
5438 decision should have priority over a permit decision. This algorithm has the following
5439 behavior.

- 5440 1. If any decision is "Deny", the result is "Deny".
- 5441 2. Otherwise, if any decision is "Indeterminate{DP}", the result is "Indeterminate{DP}".
- 5442 3. Otherwise, if any decision is "Indeterminate{D}" and another decision is "Indeterminate{P}" or
5443 Permit, the result is "Indeterminate{DP}".
- 5444 4. Otherwise, if any decision is "Indeterminate{D}", the result is "Indeterminate{D}".
- 5445 5. Otherwise, if any decision is "Permit", the result is "Permit".
- 5446 6. Otherwise, if any decision is "Indeterminate{P}", the result is "Indeterminate{P}".
- 5447 7. Otherwise, the result is "NotApplicable".

5448 The following pseudo-code represents the normative specification of this **combining algorithm**. The
5449 algorithm is presented here in a form where the input to it is an array with children (the **policies**, **policy**
5450 **sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set of
5451 obligations or advice provided by this algorithm is not deterministic.

```

5452 Decision denyOverridesCombiningAlgorithm(Node[] children)
5453 {
5454     Boolean atLeastOneErrorD = false;
5455     Boolean atLeastOneErrorP = false;
5456     Boolean atLeastOneErrorDP = false;
5457     Boolean atLeastOnePermit = false;
5458     for( i=0 ; i < lengthOf(children) ; i++ )
5459     {
5460         Decision decision = children[i].evaluate();
5461         if (decision == Deny)
5462         {
5463             return Deny;
5464         }
5465         if (decision == Permit)
5466         {
5467             atLeastOnePermit = true;
5468             continue;
5469         }
5470         if (decision == NotApplicable)
5471         {
5472             continue;
5473         }
5474         if (decision == Indeterminate{D})
5475         {
5476             atLeastOneErrorD = true;
5477             continue;
5478         }
5479         if (decision == Indeterminate{P})
5480         {
5481             atLeastOneErrorP = true;
5482             continue;
5483         }
5484         if (decision == Indeterminate{DP})
5485         {
5486             atLeastOneErrorDP = true;
5487             continue;
5488         }
5489     }
5490     if (atLeastOneErrorDP)
5491     {
5492         return Indeterminate{DP};
5493     }
5494     if (atLeastOneErrorD && (atLeastOneErrorP || atLeastOnePermit))
5495     {
5496         return Indeterminate{DP};
5497     }
5498     if (atLeastOneErrorD)
5499     {
5500         return Indeterminate{D};
5501     }
5502     if (atLeastOnePermit)
5503     {
5504         return Permit;
5505     }
5506     if (atLeastOneErrorP)
5507     {
5508         return Indeterminate{P};
5509     }
5510     return NotApplicable;
5511 }

```

5512 **Obligations** and **advice** shall be combined as described in Section 7.18.

5513 C.3 Ordered-deny-overrides

5514 The following specification defines the "Ordered-deny-overrides" **rule-combining algorithm** of a **policy**.

5515 The behavior of this algorithm is identical to that of the "Deny-overrides" **rule-combining**
5516 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL
5517 match the order as listed in the **policy**.

5518 The **rule combining algorithm** defined here has the following identifier:

5519 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-deny-overrides

5520 The following specification defines the "Ordered-deny-overrides" **policy-combining algorithm** of a
5521 **policy set**.

5522 The behavior of this algorithm is identical to that of the "Deny-overrides" **policy-combining**
5523 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL
5524 match the order as listed in the **policy set**.

5525 The **policy combining algorithm** defined here has the following identifier:

5526 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-deny-
5527 overrides

5528 C.4 Permit-overrides

5529 This section defines the "Permit-overrides" **rule-combining algorithm** of a **policy** and **policy-combining**
5530 **algorithm** of a **policy set**.

5531 This **combining algorithm** makes use of the extended "Indeterminate".

5532 The **rule combining algorithm** defined here has the following identifier:

5533 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides

5534 The **policy combining algorithm** defined here has the following identifier:

5535 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-overrides

5536 The following is a non-normative informative description of this combining algorithm.

5537 The permit overrides **combining algorithm** is intended for those cases where a permit
5538 decision should have priority over a deny decision. This algorithm has the following
5539 behavior.

- 5540 1. If any decision is "Permit", the result is "Permit".
- 5541 2. Otherwise, if any decision is "Indeterminate{DP}", the result is "Indeterminate{DP}".
- 5542 3. Otherwise, if any decision is "Indeterminate{P}" and another decision is
5543 "Indeterminate{D} or Deny, the result is "Indeterminate{DP}".
- 5544 4. Otherwise, if any decision is "Indeterminate{P}", the result is "Indeterminate{P}".
- 5545 5. Otherwise, if any decision is "Deny", the result is "Deny".
- 5546 6. Otherwise, if any decision is "Indeterminate{D}", the result is "Indeterminate{D}".
- 5547 7. Otherwise, the result is "NotApplicable".

5548 The following pseudo-code represents the normative specification of this **combining algorithm**. The
5549 algorithm is presented here in a form where the input to it is an array with all children (the **policies**, **policy**
5550 **sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set of
5551 obligations or advice provided by this algorithm is not deterministic.

```
5552 Decision permitOverridesCombiningAlgorithm(Node[] children)  
5553 {  
5554     Boolean atLeastOneErrorD = false;  
5555     Boolean atLeastOneErrorP = false;  
5556     Boolean atLeastOneErrorDP = false;  
5557     Boolean atLeastOneDeny = false;
```

```

5558 for( i=0 ; i < lengthOf(children) ; i++ )
5559 {
5560     Decision decision = children[i].evaluate();
5561     if (decision == Deny)
5562     {
5563         atLeastOneDeny = true;
5564         continue;
5565     }
5566     if (decision == Permit)
5567     {
5568         return Permit;
5569     }
5570     if (decision == NotApplicable)
5571     {
5572         continue;
5573     }
5574     if (decision == Indeterminate{D})
5575     {
5576         atLeastOneErrorD = true;
5577         continue;
5578     }
5579     if (decision == Indeterminate{P})
5580     {
5581         atLeastOneErrorP = true;
5582         continue;
5583     }
5584     if (decision == Indeterminate{DP})
5585     {
5586         atLeastOneErrorDP = true;
5587         continue;
5588     }
5589 }
5590 if (atLeastOneErrorDP)
5591 {
5592     return Indeterminate{DP};
5593 }
5594 if (atLeastOneErrorP && (atLeastOneErrorD || atLeastOneDeny))
5595 {
5596     return Indeterminate{DP};
5597 }
5598 if (atLeastOneErrorP)
5599 {
5600     return Indeterminate{P};
5601 }
5602 if (atLeastOneDeny)
5603 {
5604     return Deny;
5605 }
5606 if (atLeastOneErrorD)
5607 {
5608     return Indeterminate{D};
5609 }
5610 return NotApplicable;
5611 }

```

5612 **Obligations** and **advice** shall be combined as described in Section 7.18.

5613 C.5 Ordered-permit-overrides

5614 The following specification defines the "Ordered-permit-overrides" **rule-combining algorithm** of a **policy**.

5615 The behavior of this algorithm is identical to that of the "Permit-overrides" **rule-combining**
5616 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL
5617 match the order as listed in the **policy**.

5618 The **rule combining algorithm** defined here has the following identifier:
5619 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-
5620 overrides
5621 The following specification defines the "Ordered-permit-overrides" **policy-combining algorithm** of a
5622 **policy set**.
5623 The behavior of this algorithm is identical to that of the "Permit-overrides" **policy-combining**
5624 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL
5625 match the order as listed in the **policy set**.
5626 The **policy combining algorithm** defined here has the following identifier:
5627 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:ordered-permit-
5628 overrides

5629 C.6 Deny-unless-permit

5630 This section defines the "Deny-unless-permit" **rule-combining algorithm** of a **policy** or **policy-**
5631 **combining algorithm** of a **policy set**.

5632 The **rule combining algorithm** defined here has the following identifier:
5633 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-permit

5634 The **policy combining algorithm** defined here has the following identifier:
5635 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-permit

5636 The following is a non-normative informative description of this **combining algorithm**.

5637 The "Deny-unless-permit" **combining algorithm** is intended for those cases where a
5638 permit decision should have priority over a deny decision, and an "Indeterminate" or
5639 "NotApplicable" must never be the result. It is particularly useful at the top level in a
5640 **policy** structure to ensure that a **PDP** will always return a definite "Permit" or "Deny"
5641 result. This algorithm has the following behavior.

- 5642 1. If any decision is "Permit", the result is "Permit".
- 5643 2. Otherwise, the result is "Deny".

5644 The following pseudo-code represents the normative specification of this **combining algorithm**. The
5645 algorithm is presented here in a form where the input to it is an array with all the children (the **policies**,
5646 **policy sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set
5647 of obligations or advice provided by this algorithm is not deterministic.

```
5648 Decision denyUnlessPermitCombiningAlgorithm(Node[] children)
5649 {
5650     for( i=0 ; i < lengthOf(children) ; i++ )
5651     {
5652         if (children[i].evaluate() == Permit)
5653         {
5654             return Permit;
5655         }
5656     }
5657     return Deny;
5658 }
```

5659 **Obligations** and **advice** shall be combined as described in Section 7.18.

5660 C.7 Permit-unless-deny

5661 This section defines the "Permit-unless-deny" **rule-combining algorithm** of a **policy** or **policy-**
5662 **combining algorithm** of a **policy set**.

5663 The **rule combining algorithm** defined here has the following identifier:

5664 urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-unless-deny

5665 The **policy combining algorithm** defined here has the following identifier:
5666 urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:permit-unless-deny
5667 The following is a non-normative informative description of this **combining algorithm**.

5668 The "Permit-unless-deny" **combining algorithm** is intended for those cases where a
5669 deny decision should have priority over a permit decision, and an "Indeterminate" or
5670 "NotApplicable" must never be the result. It is particularly useful at the top level in a
5671 **policy** structure to ensure that a **PDP** will always return a definite "Permit" or "Deny"
5672 result. This algorithm has the following behavior.

- 5673 1. If any decision is "Deny", the result is "Deny".
5674 2. Otherwise, the result is "Permit".

5675 The following pseudo-code represents the normative specification of this **combining algorithm**. The
5676 algorithm is presented here in a form where the input to it is an array with all the children (the **policies**,
5677 **policy sets** or **rules**) of the **policy** or **policy set**. The children may be processed in any order, so the set
5678 of obligations or advice provided by this algorithm is not deterministic.

```
5679 Decision permitUnlessDenyCombiningAlgorithm(Node[] children)
5680 {
5681     for( i=0 ; i < lengthOf(children) ; i++ )
5682     {
5683         if (children[i].evaluate() == Deny)
5684         {
5685             return Deny;
5686         }
5687     }
5688     return Permit;
5689 }
```

5690 **Obligations** and **advice** shall be combined as described in Section 7.18.

5691 C.8 First-applicable

5692 This section defines the "First-applicable" **rule-combining algorithm** of a **policy** and **policy-combining**
5693 **algorithm** of a **policy set**.

5694 The **rule combining algorithm** defined here has the following identifier:

5695 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable

5696 The following is a non-normative informative description of the "First-Applicable" **rule-combining**
5697 **algorithm** of a **policy**.

5698 Each **rule** SHALL be evaluated in the order in which it is listed in the **policy**. For a particular
5699 **rule**, if the **target** matches and the **condition** evaluates to "True", then the evaluation of the
5700 **policy** SHALL halt and the corresponding **effect** of the **rule** SHALL be the result of the evaluation
5701 of the **policy** (i.e. "Permit" or "Deny"). For a particular **rule** selected in the evaluation, if the
5702 **target** evaluates to "False" or the **condition** evaluates to "False", then the next **rule** in the order
5703 SHALL be evaluated. If no further **rule** in the order exists, then the **policy** SHALL evaluate to
5704 "NotApplicable".

5705 If an error occurs while evaluating the **target** or **condition** of a **rule**, then the evaluation SHALL
5706 halt, and the **policy** shall evaluate to "Indeterminate", with the appropriate error status.

5707 The following pseudo-code represents the normative specification of this **rule-combining algorithm**.

```
5708 Decision firstApplicableEffectRuleCombiningAlgorithm(Rule[] rules)
5709 {
5710     for( i = 0 ; i < lengthOf(rules) ; i++ )
5711     {
5712         Decision decision = evaluate(rules[i]);
5713         if (decision == Deny)
5714         {
```

```

5715         return Deny;
5716     }
5717     if (decision == Permit)
5718     {
5719         return Permit;
5720     }
5721     if (decision == NotApplicable)
5722     {
5723         continue;
5724     }
5725     if (decision == Indeterminate)
5726     {
5727         return Indeterminate;
5728     }
5729 }
5730 return NotApplicable;
5731 }

```

5732 The **policy combining algorithm** defined here has the following identifier:

5733 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable

5734 The following is a non-normative informative description of the “First-applicable” **policy-combining algorithm** of a **policy set**.

5736 Each **policy** is evaluated in the order that it appears in the **policy set**. For a particular **policy**, if
5737 the **target** evaluates to "True" and the **policy** evaluates to a determinate value of "Permit" or
5738 "Deny", then the evaluation SHALL halt and the **policy set** SHALL evaluate to the **effect** value of
5739 that **policy**. For a particular **policy**, if the **target** evaluate to "False", or the **policy** evaluates to
5740 "NotApplicable", then the next **policy** in the order SHALL be evaluated. If no further **policy** exists
5741 in the order, then the **policy set** SHALL evaluate to "NotApplicable".

5742 If an error were to occur when evaluating the **target**, or when evaluating a specific **policy**, the
5743 reference to the **policy** is considered invalid, or the **policy** itself evaluates to "Indeterminate",
5744 then the evaluation of the **policy-combining algorithm** shall halt, and the **policy set** shall
5745 evaluate to "Indeterminate" with an appropriate error status.

5746 The following pseudo-code represents the normative specification of this policy-combination algorithm.

```

5747 Decision firstApplicableEffectPolicyCombiningAlgorithm(Policy[] policies)
5748 {
5749     for( i = 0 ; i < lengthOf(policies) ; i++ )
5750     {
5751         Decision decision = evaluate(policies[i]);
5752         if(decision == Deny)
5753         {
5754             return Deny;
5755         }
5756         if(decision == Permit)
5757         {
5758             return Permit;
5759         }
5760         if (decision == NotApplicable)
5761         {
5762             continue;
5763         }
5764         if (decision == Indeterminate)
5765         {
5766             return Indeterminate;
5767         }
5768     }
5769     return NotApplicable;
5770 }

```

5771 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

5772 C.9 Only-one-applicable

5773 This section defines the "Only-one-applicable" **policy-combining algorithm** of a **policy set**.

5774 The **policy combining algorithm** defined here has the following identifier:

5775 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable

5776 The following is a non-normative informative description of the "Only-one-applicable" **policy-combining algorithm** of a **policy set**.

5778 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".

5782 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**.

5784 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.

5787 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```
5788 Decision onlyOneApplicablePolicyPolicyCombiningAlogrithm(Policy[] policies)
5789 {
5790     Boolean          atLeastOne      = false;
5791     Policy           selectedPolicy = null;
5792     ApplicableResult appResult;
5793
5794     for ( i = 0; i < lengthOf(policies) ; i++ )
5795     {
5796         appResult = isApplicable(policies[I]);
5797
5798         if ( appResult == Indeterminate )
5799         {
5800             return Indeterminate;
5801         }
5802         if( appResult == Applicable )
5803         {
5804             if ( atLeastOne )
5805             {
5806                 return Indeterminate;
5807             }
5808             else
5809             {
5810                 atLeastOne      = true;
5811                 selectedPolicy = policies[i];
5812             }
5813         }
5814         if ( appResult == NotApplicable )
5815         {
5816             continue;
5817         }
5818     }
5819     if ( atLeastOne )
5820     {
5821         return evaluate(selectedPolicy);
5822     }
5823     else
5824     {
5825         return NotApplicable;
5826     }
5827 }
```

5828 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

5829 C.10 Legacy Deny-overrides

5830 This section defines the legacy “Deny-overrides” **rule-combining algorithm** of a **policy** and **policy-**
5831 **combining algorithm** of a **policy set**.

5832

5833 The **rule combining algorithm** defined here has the following identifier:

5834 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides

5835 The following is a non-normative informative description of this combining algorithm.

5836 The “Deny-overrides” rule combining algorithm is intended for those cases where a deny
5837 decision should have priority over a permit decision. This algorithm has the following
5838 behavior.

- 5839 1. If any rule evaluates to "Deny", the result is "Deny".
- 5840 2. Otherwise, if any rule having Effect="Deny" evaluates to "Indeterminate", the result is
5841 "Indeterminate".
- 5842 3. Otherwise, if any rule evaluates to "Permit", the result is "Permit".
- 5843 4. Otherwise, if any rule having Effect="Permit" evaluates to "Indeterminate", the result is
5844 "Indeterminate".
- 5845 5. Otherwise, the result is "NotApplicable".

5846 The following pseudo-code represents the normative specification of this **rule-combining algorithm**.

```
5847 Decision denyOverridesRuleCombiningAlgorithm(Rule[] rules)
5848 {
5849     Boolean atLeastOneError = false;
5850     Boolean potentialDeny = false;
5851     Boolean atLeastOnePermit = false;
5852     for( i=0 ; i < lengthOf(rules) ; i++ )
5853     {
5854         Decision decision = evaluate(rules[i]);
5855         if (decision == Deny)
5856         {
5857             return Deny;
5858         }
5859         if (decision == Permit)
5860         {
5861             atLeastOnePermit = true;
5862             continue;
5863         }
5864         if (decision == NotApplicable)
5865         {
5866             continue;
5867         }
5868         if (decision == Indeterminate)
5869         {
5870             atLeastOneError = true;
5871
5872             if (effect(rules[i]) == Deny)
5873             {
5874                 potentialDeny = true;
5875             }
5876             continue;
5877         }
5878     }
5879     if (potentialDeny)
5880     {
5881         return Indeterminate;
5882     }
5883     if (atLeastOnePermit)
5884     {
```

```

5885     return Permit;
5886   }
5887   if (atLeastOneError)
5888   {
5889     return Indeterminate;
5890   }
5891   return NotApplicable;
5892 }

```

5893 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

5894 The **policy combining algorithm** defined here has the following identifier:

5895 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides

5896 The following is a non-normative informative description of this combining algorithm.

5897 The "Deny-overrides" policy combining algorithm is intended for those cases where a
5898 deny decision should have priority over a permit decision. This algorithm has the
5899 following behavior.

- 5900 1. If any policy evaluates to "Deny", the result is "Deny".
- 5901 2. Otherwise, if any policy evaluates to "Indeterminate", the result is "Deny".
- 5902 3. Otherwise, if any policy evaluates to "Permit", the result is "Permit".
- 5903 4. Otherwise, the result is "NotApplicable".

5904 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```

5905 Decision denyOverridesPolicyCombiningAlgorithm(Policy[] policies)
5906 {
5907   Boolean atLeastOnePermit = false;
5908   for( i=0 ; i < lengthOf(policies) ; i++ )
5909   {
5910     Decision decision = evaluate(policies[i]);
5911     if (decision == Deny)
5912     {
5913       return Deny;
5914     }
5915     if (decision == Permit)
5916     {
5917       atLeastOnePermit = true;
5918       continue;
5919     }
5920     if (decision == NotApplicable)
5921     {
5922       continue;
5923     }
5924     if (decision == Indeterminate)
5925     {
5926       return Deny;
5927     }
5928   }
5929   if (atLeastOnePermit)
5930   {
5931     return Permit;
5932   }
5933   return NotApplicable;
5934 }

```

5935 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

5936 C.11 Legacy Ordered-deny-overrides

5937 The following specification defines the legacy "Ordered-deny-overrides" **rule-combining algorithm** of a
5938 **policy**.

5939 The behavior of this algorithm is identical to that of the “Deny-overrides” **rule-combining**
5940 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL
5941 match the order as listed in the **policy**.

5942 The **rule combining algorithm** defined here has the following identifier:

5943 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides

5944 The following specification defines the legacy “Ordered-deny-overrides” **policy-combining algorithm** of
5945 a **policy set**.

5946 The behavior of this algorithm is identical to that of the “Deny-overrides” **policy-combining**
5947 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL
5948 match the order as listed in the **policy set**.

5949 The **rule combining algorithm** defined here has the following identifier:

5950 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-
5951 overrides

5952 C.12 Legacy Permit-overrides

5953 This section defines the legacy “Permit-overrides” **rule-combining algorithm** of a **policy** and **policy-**
5954 **combining algorithm** of a **policy set**.

5955 The **rule combining algorithm** defined here has the following identifier:

5956 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides

5957 The following is a non-normative informative description of this combining algorithm.

5958 The “Permit-overrides” rule combining algorithm is intended for those cases where a
5959 permit decision should have priority over a deny decision. This algorithm has the
5960 following behavior.

- 5961 1. If any rule evaluates to "Permit", the result is "Permit".
- 5962 2. Otherwise, if any rule having Effect="Permit" evaluates to "Indeterminate" the result is
5963 "Indeterminate".
- 5964 3. Otherwise, if any rule evaluates to "Deny", the result is "Deny".
- 5965 4. Otherwise, if any rule having Effect="Deny" evaluates to "Indeterminate", the result is
5966 "Indeterminate".
- 5967 5. Otherwise, the result is "NotApplicable".

5968 The following pseudo-code represents the normative specification of this **rule-combining algorithm**.

```
5969 Decision permitOverridesRuleCombiningAlgorithm(Rule[] rules)
5970 {
5971     Boolean atLeastOneError = false;
5972     Boolean potentialPermit = false;
5973     Boolean atLeastOneDeny = false;
5974     for( i=0 ; i < lengthOf(rules) ; i++ )
5975     {
5976         Decision decision = evaluate(rules[i]);
5977         if (decision == Deny)
5978         {
5979             atLeastOneDeny = true;
5980             continue;
5981         }
5982         if (decision == Permit)
5983         {
5984             return Permit;
5985         }
5986         if (decision == NotApplicable)
5987         {
5988             continue;
5989         }
5990     }
5991 }
```

```

5990     if (decision == Indeterminate)
5991     {
5992         atLeastOneError = true;
5993
5994         if (effect(rules[i]) == Permit)
5995         {
5996             potentialPermit = true;
5997         }
5998         continue;
5999     }
6000 }
6001 if (potentialPermit)
6002 {
6003     return Indeterminate;
6004 }
6005 if (atLeastOneDeny)
6006 {
6007     return Deny;
6008 }
6009 if (atLeastOneError)
6010 {
6011     return Indeterminate;
6012 }
6013 return NotApplicable;
6014 }

```

6015 **Obligations** and **advice** of the individual **rules** shall be combined as described in Section 7.18.

6016 The **policy combining algorithm** defined here has the following identifier:

6017 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides

6018 The following is a non-normative informative description of this combining algorithm.

6019 The "Permit-overrides" policy combining algorithm is intended for those cases where a
6020 permit decision should have priority over a deny decision. This algorithm has the
6021 following behavior.

- 6022 1. If any policy evaluates to "Permit", the result is "Permit".
- 6023 2. Otherwise, if any policy evaluates to "Deny", the result is "Deny".
- 6024 3. Otherwise, if any policy evaluates to "Indeterminate", the result is "Indeterminate".
- 6025 4. Otherwise, the result is "NotApplicable".

6026 The following pseudo-code represents the normative specification of this **policy-combining algorithm**.

```

6027 Decision permitOverridesPolicyCombiningAlgorithm(Policy[] policies)
6028 {
6029     Boolean atLeastOneError = false;
6030     Boolean atLeastOneDeny = false;
6031     for( i=0 ; i < lengthOf(policies) ; i++ )
6032     {
6033         Decision decision = evaluate(policies[i]);
6034         if (decision == Deny)
6035         {
6036             atLeastOneDeny = true;
6037             continue;
6038         }
6039         if (decision == Permit)
6040         {
6041             return Permit;
6042         }
6043         if (decision == NotApplicable)
6044         {
6045             continue;
6046         }
6047         if (decision == Indeterminate)

```

```

6048     {
6049         atLeastOneError = true;
6050         continue;
6051     }
6052 }
6053 if (atLeastOneDeny)
6054 {
6055     return Deny;
6056 }
6057 if (atLeastOneError)
6058 {
6059     return Indeterminate;
6060 }
6061 return NotApplicable;
6062 }

```

6063 **Obligations** and **advice** of the individual **policies** shall be combined as described in Section 7.18.

6064 C.13 Legacy Ordered-permit-overrides

6065 The following specification defines the legacy "Ordered-permit-overrides" **rule-combining algorithm** of a
6066 **policy**.

6067 The behavior of this algorithm is identical to that of the "Permit-overrides" **rule-combining**
6068 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL
6069 match the order as listed in the **policy**.

6070 The **rule combining algorithm** defined here has the following identifier:

6071 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-
6072 overrides

6073 The following specification defines the legacy "Ordered-permit-overrides" **policy-combining algorithm** of
6074 a **policy set**.

6075 The behavior of this algorithm is identical to that of the "Permit-overrides" **policy-combining**
6076 **algorithm** with one exception. The order in which the collection of **policies** is evaluated SHALL
6077 match the order as listed in the **policy set**.

6078 The **policy combining algorithm** defined here has the following identifier:

6079 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-
6080 overrides

6081

6082

Appendix D. Acknowledgements

6083 The following individuals have participated in the creation of this specification and are gratefully
6084 acknowledged:

6085

6086 Anil Saldhana

6087 Anil Tappetla

6088 Anne Anderson

6089 Anthony Nadalin

6090 Bill Parducci

6091 Craig Forster

6092 David Chadwick

6093 David Staggs

6094 Dilli Arumugam

6095 Duane DeCouteau

6096 Erik Rissanen

6097 Gareth Richards

6098 Hal Lockhart

6099 Jan Herrmann

6100 John Tolbert

6101 Ludwig Seitz

6102 Michiharu Kudo

6103 Naomaru Itoi

6104 Paul Tyson

6105 Prateek Mishra

6106 Rich Levinson

6107 Ronald Jacobson

6108 Seth Proctor

6109 Sridhar Muppidi

6110 Tim Moses

6111 Vernon Murdoch

6112

Appendix E. Revision History

6113

6114

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>

6115